# 知的情報処理システムに関する調査研究報告書

## 第4分冊　知識システム技術者育成ガイドライン

平成元年3月

JIPDEC

財団法人　日本情報処理開発協会
ICOT-JIPDEC AI センター

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和６３年度に実施した「知的情報処理システムの導入・活用に関する調査研究」の成果の一部をまとめたものであります。

# まえがき

通商産業省情報処理振興審議会は、１９８６年３月の答申において、１９９０年代を目標とする第４期「電子計算機利用高度化計画」を取りまとめましたが、この中で、従来のコンピュータ利用技術の延長線上では捉えきれない新しい概念として「知識情報処理」を取り上げています。また、情報処理振興審議会の答申から遡ること４年、１９８２年４月に通商産業省では、世界に先駆けて、知識情報処理機能を組み込んだ全く新しいコンセプトの「第五世代コンピュータ」の開発をスタートさせました。

欧米諸国もこの日本の第五世代コンピュータプロジェクトに刺激を受け、相次いで知識情報処理技術の研究開発をナショナル・プロジェクトとして取り上げています。

一方、産業界でも、各種の知識システムの開発、導入に積極的に取り組み始めていますが、構築方法論が確立されていないことや知識獲得の問題などがあり、知識システムの多くは、未だプロトタイプの段階に留まっています。

このような現状認識に基づき、ＩＣＯＴ－ＪＩＰＤＥＣ　ＡＩセンターでは、昭和６１年度から「知的情報処理システム調査研究委員会」を設置し、学界、産業界で活躍している方々により、３年間にわたって活発な調査研究活動を行い、ここに、全５分冊に及ぶ研究成果を取りまとめました。

本調査の実施にあたっては、委員会の委員の方々はもとより関係省庁ほか関係各方面から御指導・御協力をいただきました。ここに謝意を表する次第であります。

平成元年３月

財団法人　日本情報処理開発協会

会　長　　影山　　衛司

# 「知的情報処理システムに関する調査研究」概要

わが国における知識システムの研究開発の数は 2,000を越えたとみられ，産業界のあらゆる分野への応用の裾野が広がりつつある。しかし，そのうちフィールドテストを経て，実用化の段階に達したシステムの数は，多めに見積もっても， 500以下とみられ，多くのシステムはプロトタイプの段階またはデモンストレーションの段階に留まっているとみられる。その理由として，いくつかの要因が考えられるが，その中で，特に，

1) 知識システムを構築していくことを支援する方法論が確立されていないこと，

2) 知識システムを試作・実装するためのツールが十分には整備されていないこと，

3) 知識獲得を支援する方法論やツールの研究開発が立ち遅れていること，

4) 知識システム技術者が不足し，育成の指針が確立されていないこと，

5) 開発事例を蓄積・分析し，結果を有効に利用する体制が確立されていないこと，

などが指摘されている。このような現状認識に基づき，ICOT-JIPDEC AIセンターは昭和61年9月に「知的情報処理システムに関する調査研究委員会」を設置し，つぎのようなテーマに関する調査研究活動を行ってきた。

1) 知識システム構築方法論を確立すること，

2) 知識システム構築ツールの評価体系を確立すること，

3) 知識獲得支援の動向調査を行い，今後あるべき支援のイメージを提案すること，

4) 知識システム技術者を育成するためのガイドラインを作成すること，

5) 知識システム開発事例の収集・分析を行い，今後の開発の資とすること，

調査研究活動の前半（昭和61年9月～昭和62年9月）では，知識システム構築方法論および知識システム構築ツールの評価に重点をおき，その研究成果は昭和62年9月の成果発表会で報告され，その詳細は「知識システム開発方法論」および「知識システム開発ツールの評価」と題する報告書にまとめられている。

調査研究活動の後半（昭和62年9月～平成元年3月）では，計画・設計型問題の構築方法論，構築ツールの性能評価，知識獲得支援の動向分析，知識システム技術者の育成方法および知識システム開発事例の分析に重点をおいて活動を展開してきた。

調査研究の成果は，以下の5分冊からなる本報告書にまとめられている。

第1分冊「計画・設計型知識システムの構築方法論」では，計画・設計型知識システムの特徴分析を行い，計画・設計型問題には探索的側面，意思決定的側面，事例利用的側面および知識獲得的側面の4つがあることを指摘，各側面に対して，問題解決的接近，仮説推論的接近，

事例ベース推論的接近および学習的接近の必要性と枠組みをまとめている。

第２分冊「知識システム構築ツールの評価」では，システム構築ツールを評価するための評価体系を整備，ツールの性能評価を行うための標準問題を作成，ＡＩセンターオープンハウスに導入されているツールに対し性能評価のテストを行い，評価結果を取りまとめるとともに，ツールの評価に関するいくつかの提言を行っている。

第３分冊「知識獲得支援」では，知識獲得支援ツールの位置づけ・分類を行い，文献調査とヒアリングに基づき，各ツールの機能と特徴を客観的にとりまとめ，知識システム開発事例における知識獲得の実際を分析，知識獲得支援のツールのあり方および知識獲得支援ツールに対する要求事項をまとめている。

第４分冊「知識システム技術者育成ガイドライン」では，知識システム技術者の役割を，知識システムの開発手順の段階に対応して，システムアナリスト，プロトタイピングアナリスト，知識プログラマ，保守・管理技術者の４つに類別した上で，修得すべき技術を人工知能技術，知識工学技術，システム技術の３つに分け，その概要をまとめている。

第５分冊「知識システムの事例」では，生体や社会などの解釈／診断問題事例，プラントや人工衛星などの制御問題事例，スケジューリング，レイアウト，ＬＳＩ設計などの計画／設計問題事例など全部で２３の代表的な事例を取り上げ，比較可能な同一の枠組みで紹介し，さらに各システムにおける基本タスクと問題解決機能を明らかにしている。

以上を要するに，本報告書は知識システムの研究開発においてその解決が緊急とされる課題を重点的に取り上げ，産業界と学会で活躍している技術者・研究者からなる横断的な組織によって調査研究を行った成果である。本報告書がわが国における知識システム研究開発の一層の発展に寄与する資として役立つことを願っている。

最後に，本調査研究を取りまとめる上で，数十回におよぶ会合を通じて，終始，熱心にご協力頂いた委員の各氏ならびにこれを支えてくれた事務局の皆様に感謝の意を表する。

平成元年３月

知的情報処理システム調査研究委員会　委員長　小林重信

# 昭和６３年度　知的情報処理システム調査研究委員会

## 委　員　名　簿

| | | |
|---|---|---|
| 委員長 | 小　林　重　信 | 東京工業大学 |
| 幹　事 | 寺　野　隆　雄 | (財)電力中央研究所 |
| 委　員 | 関　根　史　麿 | 花王(株)　知識情報科学研究所 |
| 〃 | 菊　地　一　成 | キヤノン(株)　情報システム研究所 |
| 〃 | 山　崎　知　彦 | (株)豊田中央研究所 |
| 〃 | 森　　啓 | 日本電気(株)　Ｃ＆Ｃシステム研究所 |
| 〃 | 千吉良　英　毅 | (株)日立製作所　システム開発研究所 |
| 〃 | 中　島　俊　哉 | 富士通(株)　科学システム部 |
| 〃 | 小　林　慎　一 | (株)三菱総合研究所 |
| 〃 | 福　島　正　俊 | 三菱電機(株)中央研究所 |
| 〃 | 森　谷　正　晴 | 新日本製鐵(株)　君津製鐵所　設備部 |
| 〃 | 桃　井　茂　晴 | 日本電信電話(株)　ＮＴＴ情報通信処理研究所 |
| 〃 | 畝　見　達　夫 | 長岡技術科学大学 |
| オブザーバ | 生　駒　憲　治 | (財)新世代コンピュータ技術開発機構 |
| 〃 | 渡　辺　　敏 | 新日本製鐵(株)　君津製鐵所　設備部 |
| 事務局 | 平　井　吉　光 | (財)日本情報処理開発協会　ＡＩ振興センター |
| 〃 | 茂　呂　知　明 | 〃　　〃 |

# 知識システム技術者育成ガイドライン・目次

# １．育成指針の概要

# 1. 育成指針の概要

本章では，まず知識システムに対する基本的な理解を得るために，知識システムの特徴を述べる．つぎに，知識システムの開発の手順を示し，その中で知識システム技術者に課されるタスクと役割を述べる．さらに，そのような役割を遂行するうえで，知識システム技術者が修得することが必要なまたは修得することが望ましい技術の概要を述べる．

## 1.1 知識システムの特徴

知識システムは，一般に，つぎのような特徴をもつ．

### 1)悪構造・悪定義問題を対象とすること

機能や構造が明確であり，数理的あるいはアルゴリズム的な問題解決が可能な問題は良構造(well-structured)であるといわれる．また要請される問題解決の範囲を事前に明確に設定することができる問題は良定義(well-defined)であるといわれる．これに対し，アルゴリズム的な問題解決が困難な問題は悪構造(ill-structured)，また問題解決の範囲を事前に明確に設定することが困難な問題は悪定義(ill-defined)であるといわれる．知識システムは，一般に，悪構造・悪定義問題を対象とする．

### 2)プロトタイピング的接近を必要とすること

悪構造・悪定義問題では，問題解決の範囲（ユーザの要求仕様）および問題解決の方法を事前に明確化することが困難である．取り敢えず，動作可能なプロトタイプシステムをつくることにより，ユーザの要求仕様が次第に明らかになったり，あるいは問題解決の手掛りを見出したりすることが少なくない．したがって，知識システムの構築にあたっては，プロトタイピング的接近を何回か繰り返す方式が有効な場合が少なくない．

### 3)人工知能や知識工学を基盤技術とすること

知識システムにおける問題解決の方法論的基礎は人工知能に求めることができる．また知識システムを構築していく方法論的基礎は知識工学に求めることができる．人工知能は，探索や問題解決戦略，知識の表現と利用などの技術を提供する．一方，知識工学は構築方法論，知識獲得支援，知識プログラミングなどの技術を提供する．しかし，人工知能や知識工学を補完する技術として，システム技術も重要である．

4)知識処理を中心的なタスクとすること

知識システムは，対象とする問題に関するデータが格納されるワーキングメモリ，問題解決に必要な知識が格納される知識ベースおよび問題解決のプロセスを制御する推論機構から構成されるので，知識の表現，推論の制御，知識の獲得と管理など知識処理が問題解決において主要な役割を演じる．

5)システムの透明性が要請されること

プロトタイピング的方法に基づいて知識システムを段階的に開発するには，システムがブラックボックスであってはならず，システムの透明性(transparency)が要請される．ま知識システムと推論機構が分離されていることは，システムの透明性を実現するうえでの必要条件である．しかし複雑な問題解決が要請されるときには，メタ知識と呼ばれる推論を間接的に制御するための知識を知識ベースにおかざるを得ないこともあり，システムの透明性を確保するには，十分な配慮が必要である．

6)ユーザとの対話性を向上させること

知識システムはユーザにとってもブラックボックスであってはならない．ユーザに対し，推論結果を示すだけではなく，結果に至る推論の筋道をその根拠とともに明確に示すことにより，ユーザとシステムの間の対話性を向上させる必要がある．

7)保守や拡張が容易に行えること

知識システムは，一般に，開いた領域を対象とするので，システムとして完成したと呼ばれる段階に達するのは稀であり，言換えれば，知識システムは永遠にプロトタイプシステムであり続けるともいえる．この場合，知識システムの保守と拡張がシステム開発者からユーザに円滑に委ねられることが望ましい．すなわち，知識システムは手離れの良いシステムであることが望ましい．そのためにも，システムの透明性やユーザとの対話性の確保は重要である．

## 1.2 知識システムの開発手順

知識システムの開発は，1)システムアナリシス，2)プロトタイピングアナリシス，3)知識プログラミング，4)保守・運用，の4段階に大別される．図1に知識システムの開発手順を示す．

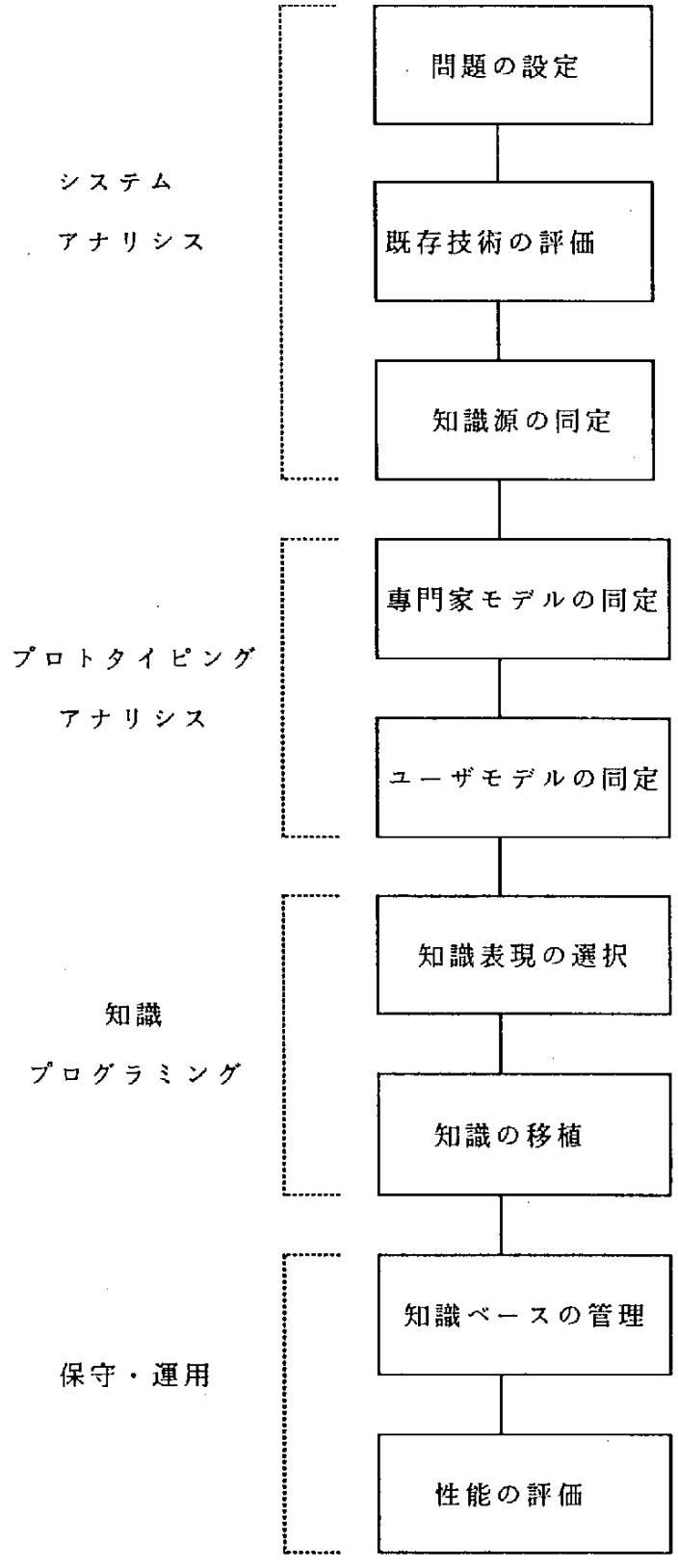


図1　知識システムの開発手順

## 1.3 知識システム技術者の定義と役割

ここでは，知識システム技術者を，つぎのように定義する．

「知識システム技術者とは，知識システムの研究開発プロジェクトを実施するうえで，領域専門家およびユーザの協力のもとに，システムの計画・設計・構築・運用・保守に伴う一連のタスクを実際に行うことができるものをいう」

この定義に基づいて，図1に示される知識システムの開発手順に即して，知識システム技術者に要請されるタスクとその役割を述べることとする．

### 1) システムアナリシス

システムアナリシスの目的は，対象とすべき問題を設定し，既存技術との比較において知識システム的接近の必要性を明らかにした上で，問題解決に必要な知識源の所在を同定することにある．

①問題の設定

知識システム開発の必要性および十分性をチェックした上で，対象の候補とされる問題領域より問題を切り出し，定義すること．

②既存技術の評価

従来技術（システム技術やソフトウエア技術）との比較分析を行い，知識システム的接近の妥当性や優位性を明らかにすること．問題によっては，既存技術との融合形式もあり得ることを示す必要がある．

③知識源の同定

問題解決に必要な知識が存在する知識源の所在を同定し，各種知識源に保有されている知識の形態，信頼性，利用可能性などを評価すること．領域専門家は有力な知識源であるが，唯一のものではない．教科書，設計仕様書，操作手引書，保守記録書なども有用な知識源である．

### 2) プロトタイピングアナリシス

プロトタイピングアナリシスの目的は，領域専門家との対話を通じて専門家モデルを明かにした上で，問題解決に必要な知識源の所在を同定することにある．

①専門家モデルの同定

プロトタイプ開発を通じて，専門家がカバーする知識の範囲，知識の質と量，問題解決の基本制御ループ，推論の方法などを明らかにすること．

②ユーザモデルの同定

プロトタイプ開発を通じて，ユーザの要求仕様を顕在化させるとともに，ユーザの使い方を反映したユーザインタフェースを明らかにすること．

3）知識プログラミング

知識プログラミングの目的は，知識の表現および知識の利用の枠組みを決定し，知識源に存在する知識を抽出し，これを知識ベースに移植することを目的とする．

①知識表現の選択

問題解決にとって有用と考えられる知識の表現および知識の利用の枠組みを検討し，最適な方式を選択すること．

②知識の移植

各種知識源に存在する知識をシステマティックに抽出し，これを利用可能な形式に変換して，知識ベースに移植すること．

4）保守・運用

保守・運用の目的は，知識システムの評価を行い，重要な情報をドキュメント化し，さらに保守や拡張の方法を明らかにしておくことにある．

①システムの評価

知識システムの機能的・性能的評価について，知識ベースの完全性や無矛盾性，推論結果の妥当性，推論速度，メモリ容量，対話性などを評価すること．

②ドキュメンテーション

ユーザ向けに，利用範囲，使い方などを，技術者向けに，知識源情報，問題解決の方式，推論方式，メタ知識，高速化技術などをドキュメント化すること．

③保守・拡張

知識システムの保守や拡張の方法および注意事項を，ユーザ向け／技術者向けに記述しておくこと．

## 1.4 知識システム技術者が修得すべき技術

知識システム技術者が修得しておかなければならない技術は，

1)人工知能技術，　2)知識工学技術，　3)システム技術，

の３つに大別される．人工知能は知識システムによる問題解決の基礎を与える技術であり，知識工学は知識システム構築の方法論を与える技術であり，システム技術は人工知能や知識工学を補完するとともに，従来技術と知識処理技術の統合を図る上で必要とされる技術でもある．以下に，知識システム技術者が修得することが望まれる技術の概要を示す．

### 1) 人工知能技術

人工知能技術は，知識システムによる問題解決および知識処理の基礎を与える技術であり，人工知能基礎および知識処理技術に大別される．

①人工知能基礎

ＡＩ的問題解決の基礎を与える探索や問題解決戦略，知識の表現と利用の基礎を与える論理と推論および知識と表現，演繹的推論を補完する高次推論，知識獲得の基礎を与える学習などに関する技術．知識処理では対処することが 困難な認知処理の基礎を与えるニューラルネットなどに関する技術も修得しておくことが望ましい．

②知識処理技術

知識システム構築の中核をなす知識処理について，知識表現，推論制御，知識獲得，知識ベース管理などに関する技術．

### 2) 知識工学技術

知識工学は，知識システム構築の方法論を与える技術であり，構築方法論，知識獲得支援，知識プログラミング，プロトタイピングなどを主たる内容とする．

①知識システムの構成

知識システムに要請される条件，知識システムの構造と機能および知識システムの動作原理などに関する技術．

②構築方法論

知識システム開発の手順，問題の類型とタスク，解釈・診断・制御など解析型問題への特徴と接近法，計画・設計など合成型問題への特徴と接近法などに関する技術．

③知識獲得支援

知識システムの開発段階に応じた知識獲得問題，知識獲得支援のための技法，知識獲得支援ツールなどに関する技術．

④知識プログラミング

プログラミング言語，知識システム開発ツール，従来技術との接続，ユーザインタフェースなどに関する技術．

⑤プロトタイピング

スフトウエア開発のライフサイクル，プロトタイピングの型およびプロトタイピングのサイクルに関する一般的な知識ならびに知識システム開発におけるプロトタイピングなどに関する技術．

⑥知識システムの評価・運用

プロジェクトの管理，知識システムの評価，知識システムの運用，知識システムの保守などに関する技術．

3) システム技術

システム技術は人工知能技術や知識工学技術を補完するとともに，知識システムを従来技術でつくられた既存システムに接続する上で必要とされる技術でもある．

①システム基礎技術

数理工学，システム工学，ソフトウエア工学，コミュニケーション技法などに関する技術．

②システム化技術

計算機システム，データベース，通信システム，フーマンインタフェースなどに関する技術．

③システム開発技術

システム分析と要求定義，システム設計，プログラム作成技法，テスト・検査，システム評価，保守・運用などに関する技術．

以上，修得が必要な技術の概要を述べた．なお，各技術の詳細は2章～4章に示す．

（1章担当 小林重信）

# 2．人工知能技術

# 2．人工知能技術

## 2.1　人工知能基礎

### 2.1.1　探索

知識システムでは、解を直接的に手続きで求める手段がなく、何らかの方法で解を探す必要がある。ここでは、最も基本的な3種類の探索方法について述べる。

キーワード：生成検査法，盲目的探索，最良優先探索

生成検査法(generate-and-test)

本手法は，1)可能な解の生成，2)解の1個を選択し，目的とする解との比較，3)目的とする解であれば終了，そうでなければ1)に戻る，との3段階で解を探索する手法である．可能な解の生成が系統的に行われれば，解が存在する限り解を見出すことができる特徴がある．また，2)の段階で完全な解が生成されなければならないので，縦型探索である．なお，問題空間が極めて大きい場合には時間がかかる，解を無作為に生成すれば解がすぐには見つからない等の欠点がある．

盲目的探索(blind search)

解を求める探索において，探索経路を考慮する順序が任意で，解がどのあたりにありそうかの判断に，問題領域固有の情報をまったく用いない手法である．解の経路をこれの全てを含むに足る状態空間グラフで表現した場合，横型探索では，開始節点から，各節点に到る枝の個数によって測られる距離の短い経路から探索する手法である．解が存在する場合，最短パスの解を見出すことができるが，解節点の深さが深くなると生成すべき節点数は急増する欠点がある．また，縦型探索は，最も深い節点を最初に探索する手法である．次に探索する経路は1個ですむが，最初に見出した解が最短パスの解とは限らないという欠点がある．

最良優先探索(best-first search)

盲目的探索では、次に探索する節点は深さ・幅方向に対して任意であるが，この方法で

は，探索の各段階で，それまでに生成した次節点の中で最も有望な節点を適切な発見的評価関数を適用し選び出して探索する．評価関数の利用方法には多種ある．通常は，横型探索に対して評価関数を適用して実現され，常に評価値が最良になるように，また，ある枝の探索が失敗した場合，次に評価値の高い枝を探索するので，解が見つかった場合は最良のものである．分岐限定法もこの探索法の一種であり，評価関数により，それ以上探索しても解の存在する可能性のない枝を刈り込む方法である．

### 2.1.2　問題解決

一般的に知識システムの対象とする問題の全探索空間は膨大なため，問題全体をいくつかの部分問題に分割して効率的な探索する方法が必要である．ここでは3種の探索を効率化する方法について述べる．

キーワード：トップダウン精密化，拘束最小化，階層的生成検査法

トップダウン精密化(topdown refinement)

生成検査法で解の構成の詳細度・精度に着目して階層化し，生成する可能な解の数の増大を防ぐ手法である．まず，最上位の階層において拘束条件の優先度に着目して，優先度の高い条件を先に満足させるよう粗い精度の構成で解を求め，次にこの結果をより下位の階層での制約条件として伝播し，次の階層でより構成を詳細化・精密化していく．なお，上位階層から下位階層への制約条件の伝播はトップダウン的に行われ，階層間のバックトラックは行われない．解空間が包含関係で階層化できることが必要である．制約条件が比較的緩く，解の最適性があまり要求されない場合有効である．

拘束最小化(least commitment ；最小拘束，最小委託)

生成検査法で解の構成の詳細度・精度に着目して階層化した場合，上位階層での決定が階階層で生成する解や実現可能性を妨げないよう，下位階層に伝播する制約条件を最小にしておく手法である．上位階層から伝播された制約条件はもう捨てられることがないという保証が得られた場合のみ詳細化される．解空間が包含関係で階層化できることが必要である．制約条件が比較的厳しい場合に有効である．

階層的生成検査法(hierarchical generate-and-test)

生成検査法を階層的に複数適用し，各階層で生成する可能な解の数を増大を防ぐ手法である．まず，最上位の階層において生成検査法により解を求め，次にこの結果をより下位の階層の制約条件として伝播し，次の階層での生成検査法を適用していく．

なお，途中の階層までで最終解が見つからないまま探索すべき枝が無くなった場合には，上位の階層に戻って探索を探索する．解空間が階層的に分割できることが必要である．制約条件が比較的厳しい場合に有効である．

### 2.1.3　論理と推論

論理による表現は形式化するのが容易であり，厳密であること，推論を行う処理とは独立でその使われ方を考慮しなくてもよいこと，モジュール性が高いこと，等の利点を持つものである．ここでは，基本的な論理による表現と，推論について述べる．

キーワード：命題論理，１階述語論理，Horn論理，反駁証明，導出原理

命題論理(propositional calculus)

論理学の最も基本的な概念である真(TRUE)と偽(FALSE) という２つの異なる真理値を用いた論理体系を用いて知識を表現するものである．命題(proposition) とは真あるいは偽のどちらかを取る文で，And,Or,Not,Implies（含意，⊃，→）,Equivalent 等の文連結詞を使って単純な命題を組み合わせて関係や含意を表現する．また，一般的な推論規則としては三段論法があり「ＸとＸ→Ｙがという２つの命題が真であれば，Ｙが真であることを推論できる」というもので，前もって与えられたいくつかの命題から１つの新しい命題を演繹でき，元の命題が真であればこの新しい命題も真であることが保証される特徴がある．

１階述語論理(first order predicate calculus)

命題論理の概念の拡張であり，文連結詞の意味はそのままで，述語（predicate)と関数(function)の考え方を導入した論理体系である．述語論理では変数と∀（全ての）と∃（存在する）の概念が加えられている．また，述語とは，特定の対象物（individual，個体）自身と，複数のそれらの関係に関する記述のことで，決まった個数の引数に適用され，その引数として特定の対象物が用いられた場合，真・偽のいずれかの値を取るものである．

なお，述語の引数の値のとり得る値が項のみで述語をとらないので「一階」と呼ぶ．さらに，関数とは，真・偽の値を取るのみでなく，「その引数に関連した値を返す」との概念で，その引数には変数，定数，関数がなり得，関数を結合することも可能である．

Horn論理(Horn logic)

1階述語論理のサブセットでHorn節(Horn-clause) のみで記述され，導出による機械的な計算のより高速化を実現するための論理体系である．なお，Horn節とは，Not(否定，¬)を高々1つしか含まない節である．Horn論理はその計算の高速性からPROLOG言語に利用されている．

反駁証明(refutation)

記述を証明する（記述が正しいことを調べる）のに，記述の否定が，他の既に明らかになっている記述と矛盾することを用いて証明すること．

導出原理(resolution principle)

別項の反駁証明を用いて一般的な命題の値を機械的に計算する方法である．
具体的には，①与えられた「真」の命題を節形式に変換する〔全ての項がOr（∨，または）で結合され，それぞれの項の中にはAnd(∧，かつ）やOrを含まない単純な命題，あるいはその否定形〕，②証明すべき命題の「否定」を節形式にして加える，③②の結果から矛盾を導く（結果として「空」が得られる），の手順である．

2.1.4　知識と表現

知識システムで取り扱う知識はどんな情報でどの様に分類できるか，また，これらの知識の表現方法としてどの様な分類があるかについて，知識の分類法について3種，知識表現の分類法について2種述べる．なお，実際の知識システムでの知識表現としては，宣言的表現と手続き的表現とが組み合わされていることが多い．

キーワード：領域知識と制御知識，深い知識と浅い知識，機能的知識と構造的知識，
手続き的表現，宣言的表現

領域知識と制御知識(domain knowledge and control knowledge)

領域知識とは，問題領域固有の情報であり，その問題を解くのに必要な，問題分野における概念，概念の構成・構造，概念間の相互関係，および，これら構成・構造・相互関係を作り出す・発見する方法，等である．

通常，領域知識のみをいれた知識システムでは推論がうまく動作せず，何らかの推論の制御が必要である．これが制御知識である．なお，別な観点からは，領域知識を利用して，問題領域におけるある結果を得るための方法の情報であり，領域知識に対してはより上位レベルの情報の，メタ知識であるともいえる．

深い知識と浅い知識(deep knowledge and shallow knowledge)

深い知識とは，問題領域の一般原理や対象の詳細な知識，理論的知識に代表される知識で，問題領域を根本的に理解するために必要であり，また，専門家が未経験な状況に出会って熟考する時に用い，原理的には多様な解釈が可能である．

浅い知識とは，専門家の経験的知識に代表される知識で，問題を効率良く解決できるよう整理されており，特定の型の問題解決に特化されており，想定される状況に対しては有効に働くが，そうでない場合には全く応用が効かない，また，１通りの解釈しかできず，柔軟性に乏しい．

機能的知識と構造的知識(functional knowledge and structral knowledge)

機能的知識とは，知識の機能面に着目した分類である．その構造や実現方法等を考慮せず，抽象的なブラックボックスとしてとらえたものである．構造的な知識よりも高いレベルの知識ともいえる．具体的には，装置やソフトウェアの機能，外部仕様等がこれに該当する．

構造的知識とは，知識の構造面に着目した分類である．具体的には，装置やソフトウェアのモジュール構成，モジュール間接続等がこれに該当し，推論で直接参照・操作されることの多い対象にもなる．

手続き的表現(procerual representation)

知識の動的(dynamic) 側面に着目した表現方法で，例えば，「もし～であれば～する」等の表現がこれに該当する．動作とか事柄の関係を手続きとして記述するのに便利，２階

の知識や発見的知識の埋め込みが容易，領域依存の問題解決に適する等から，例えば，専門家の持つ経験的知識などの浅い知識の記述に向く特徴があるが，知識の多面的な利用での柔軟性に劣る．具体的な例としては，OPS5等がある．

宣言的表現(declarative representation)

知識の静的(static)側面に着目した表現方法で，例えば，「～は～であれば」等の表現がこれに該当する．事実を宣言的に記述するのに便利，知識の追加・変更が手続き的表現に比べて容易，表現構造がが単純で理解しやすい，形式推論が容易，モジュール性が高い等から，知識の多面的な利用での柔軟性が高く，例えば，設計対象のモデルの性質の記述，概念，物などの構造的知識・深い知識の表現に向く特徴がある．具体的な例としては，フレーム，意味ネットワーク，スクリプト等がある．

〔2.1.1～2.1.4節担当　桃井〕

### 2.1.5 高次推論

現実に人間の行なう推論は，伝統的な形式論理の枠組みでは表現しきれないものが多い．たとえば人間は，あいまいな知識，時間によって変化する事実，常識的知識などを日常的に，しかも実に巧妙に使っている．このメカニズムを機械の上に実現する枠組みとして考え出されたのが高次推論と呼ばれる様々な手法である．

キーワード：不確実性推論，ファジイ推論，仮説推論，デフォルト推論，定性推論，協調型推論，時制推論，事例ベース推論

不確実性推論 (Uncertainty Reasoning)

医療診断などの分野では，対象システム（人体）の構造が明らかでないため，診断知識は不確実性を伴うこととなる．このような知識に基づく推論の手法として，知識をルールに表現し，ルール毎に確実性の度合いを表わす数値を割り付け，その値をもとに推論結果の確実性の度合いを計算する方法がある．ＭＹＣＩＮでは確信度つまりＣＦ(Certainty Factor)値による方法が採られた．ＣＦ値はルールから導き出される全ての仮説に割り振られる．値は -1から１の間の連続値を取り，１はその仮説を信じる根拠があることを，-1は否定する根拠があることを表わす．また，Dempster-Shafer理論に基づくものや，N. Nilsonらによる確率論理に基づく推論方式も提案されている．

ファジイ推論 (Fazzy Reasoning)

あいまいな規則をもとに，主に制御のための連続値を扱う推論を行なう場合に用いられる手法である．前項の不確実性推論の手法とは異なり，複数のルールの連鎖による多段階の推論は行なわず，入力された条件に適合する複数のルールの結論を合成し，結果を得るものが一般的である．各ルールの条件部は状況を表わす記号表現の論理結合からなる．個々の記号表現は状況に関するあるパラメータの度合いを分類する形容詞を伴うものがよく用いられ，具体値との適合の度合いを表わすメンバシップ関数が定義される．条件の満足の程度はファジイ論理に従って計算される．個々の真偽値は０から１の間の連続値をとる．０は偽，１は真に対応する．論理和では最大値を，論理積では最小値の結果の真偽値とする．ルールの結論部は目標となる操作量のメンバシップ関数である．複数のルールから得られた結論は，それらの真偽値に基づく加重平均を求めることによって合成される．

仮説推論 (Hypotheses Reasoning)

論理に基づく推論において，事実と規則の他に仮説を立てて，ある事実の証明を試み，失敗すれば仮説を棄却し，成功すれば採択するという考え方に基づいた手法である．D.L. Pooleらの Theorist は仮説推論を診断に応用したシステムである．複合的な原因による症状からの診断に応用できる．故障箇所あるいは病名を仮説とし症状の証明を試みる．また，J.J.Fingerらの RESIDUE は設計問題を扱う．設計仕様を仮説とし，要求仕様の証明を試みる．設計問題では一般に解空間が広いため，仮説の生成方法に工夫がなされている．また，仮説を管理する方法として，J.Doyle のＴＭＳおよびJ.DeKleerのＡＴＭＳがある．これらは仮説を支持する根拠の変化に伴って個々の仮説を採択あるいは棄却するメカニズムである．複数の複雑な仮説を同時に表現する方法としては多重世界表現がある．

デフォルト推論 (Default Reasoning)

明示的な知識が与えられていない場合に，なんらかの結論を導くための手法である．例えば「特に断わりがない限り鳥は飛べる」といった知識を用いる．Reiter の形式化によれば「¬Fly(x)が証明されない限り∀x.Bird(x)⊃Fly(x)である．」となる．Tweety が鳥であることが分っているとき，Tweety が飛べないことが結論できなければ，飛べると結論する．ペンギンやダチョウは鳥ではあっても飛べないから，そのことを明示的に知識として記録しておく必要がある．Tweety がペンギンなら飛べないことが結論されるが，同時にTweety は鳥であるという知識から飛べることも結論される．このように矛盾が生じる場合は，より特殊な知識，この場合には鳥よりもペンギンに関する知識を優先し，Tweetyは飛べないとする結論を採用する．

定性推論 (Qualitative Reasoning)

連続量を状態パラメータの値として持つような対象について，定量的な値を導く方程式は不明だが定性的な規則が分っている場合，大まかな予想を行なう場合，説明の生成などに用いられる．状態の因果関係に関する定性的な規則に基づく推論を行なう．このような知識の表現方法として J.DeKleerらの定性微分方程式，K.D.Forbusの定性的プロセス理論などがある．パラメータ値は適当に分割された区間に分類され区間のラベルを値とする．符号に関する分割 {+,0,-} がよく用いられる．予想問題では定量的な情報を落としてしまうため，あいまいさが生じ，複数の結論が得られる場合がある．

協調型推論 (Co-operative Reasoning)

異なる複数の知識源の各々が適当に分割された副問題を解き，相互に通信しながら全体の問題を解く．既存の知識源を組合せる場合，物理的に離れた場所にある知識源を使う場合，知識記述のモジュール性を重視する場合などに用いられる．協調問題解決のモデルとしては，Hearsay-Ⅱや AGE などに用いられた黒板モデル，R.G.Smithの契約ネットモデル，久野らの参照モデル，北村らの回覧板モデルなどがある．問題の分割，知識の分散，知識源間の制御の構造，知識源間の通信と同期，個々の知識源の推論メカニズム，協調のための推論メカニズムなどが実現上の問題点となる．

時制推論 (Temporal Reasoning)

対象の状態が時間経過に従って刻々と変化する場合に用いられる．計画問題や制御問題に特徴的である．状態，事象，行為を時間軸の上に位置付けた表現を用いる．D.McDermottの提案による，時間軸に添って連続する状態の列による表現方式，J.F.Allenによる，時区間を中心にした表現方式などがある．自然言語の意味表現を目標とした時制論理とは関連性は深いものの研究の意図する方向性が異なる．

事例ベース推論 (Case-Based Reasoning)

全ての場合に適用し得る一般的知識は明確でないが，典型的と思われる具体的な問題解決過程の事例が存在する場合に用いられる．成功あるいは失敗の事例をインデキシング等の手法により検索しやすい形式で蓄積しておき，新たな問題解決に利用する．事例の表現，組織化，検索，変更，獲得などの方法が問題となる．類似の問題を何度も繰返し解くようなシステムでは，事例を参照することにより推論の効率を上げることも考えられる．

2.1.6 学習

学習とは，システム自身が自らの経験をもとに自らの性能を向上させる機能である．広い意味ではデータの暗記も学習の一部であるが，自身の知識の構造を変化させるものが，特に学習メカニズムとして取り上げられている．

キーワード：帰納的学習，演繹的学習，発見的学習

帰納的学習 (Inductive Learning)

複数の例から，主に構文的な類似性に注目して一般規則を見出し，未知の問題に適用する．最も単純なものは分類規則の学習である．すなわち，あるカテゴリ(目標概念)についてそれに含まれる例(正の例)と含まれない例(負の例)を幾つか与え，未知の例についてそのカテゴリに含まれるかどうかを予測させるといったものである．具体的には，全ての正の例を含み，全ての負の例を排除するようなパターン記述を生成し，未知のデータについて，そのパターンと照合すれば目標概念に含まれ，照合しなければ含まれないと予想する．パターン記述言語としては，数値を含むデータを対象とする場合には，区間や，数値間の関係を表す方程式など，文字列や画像の場合には様々な文法，述語集合では，変数を含む論理式などが用いられる．パターン生成の方法として T.Mitchell の候補削除アルゴリズムがある．適切なパターン記述言語の設計が困難な場合や，入力データに雑音を含む場合には，統計的な情報や経験則に従ったもっともらしい推論が必要となる．

演繹的学習 (Deductive Learning)

説明に基づく学習(ＥＢＬ)では，豊富な領域理論を元に単一の例から目標概念の分類規則を得る．領域理論とは教科書的知識，すなわち十分な情報を含んではいるものの，問題解決のための制御情報がないために，それだけでは例題の解法が不可能な知識のことである．例題とは目標概念を導くような状況の１つであり，説明とはその状況と領域理論を使って目標概念を導き出す論理証明木である．この証明木を一般化し操作性規範を満足する一種のマクロルールを得る．操作性規範とは後の推論での新ルールの使用可能性保証する判定基準である．その他に例題を通して問題解決の実行効率を向上させるメカニズムとしてルールのチャンキングがある．

発見的学習 (Discovery)

発見システムは，種となる知識を様々に組合せることによって新たな知識を作り出し，与えられた興味深さの評価基準を元に，ある程度以上の評価を得た新知識を提示する．D.B.Lenat のＡＭは，集合論の公理を種に仮説としての様々な定理を作り出し検証することを繰返すシステムである．各仮説はタスクと呼ばれる実行単位に割当てられ，実行可能なタスクが交互に実行される．このとき重みの大きいタスクほど多くの計算時間が割当てらるため，結果として最良優先探索が進められる．生成した定理の証明は行なわないが，十

分な量の例あるいは典型的な例について検証を試みるので間違いを犯すことは少ない．実際の試行結果では，自然数，偶数，奇数，素数などの概念を発見した後，Goldbach の仮説(2より大きな全ての偶数は二つの素数の和になる)を発見した．

2.1.7 ニューラルネット

ニューラルネットは，人間の脳を中心とする神経回路網を構造的に模倣した計算モデル，あるいは，比較的単純な計算能力しか持たない素子を多数ネットワーク状に結合した計算機構である．並列計算機の実現により，現実問題に応用可能な手法として注目を集めている．

キーワード：Hopfieldモデル，逆伝播学習，競合学習

Hopfieldモデル (Hopfield Model)

Hopfieldのネットワークの構造は次のとおりである．個々のユニットは状態として1または0を取り，多入力1出力で，入力の加重和が出力となる．ネットワークのトポロジカルな構造には特に規定はない．ユニットiの時刻tにおける状態を$u_i(t)$とすると，次の時刻 t+1 の状態$u_i(t+1)$は，もし$\Sigma w_{ij} u_j(t) + s_i$が閾値$\theta_i$よりも大きければ1，小さければ0となり，等しい場合は変化させない．ここで，$w_{ij}$はユニットjからユニットiへの入力の重み，$s_i$はネットワークの外部からの入力である．各入力の重みを適当に設定することにより，パターンの連想記憶を形成することができる．各ユニットの状態を全て合せたものをネットワークの状態と考えて，次の式で表わされる値をネットワークのエネルギーと呼ぶ．

$$E(\alpha) = \Sigma_i \Sigma_j w_{ij} u_i(\alpha) u_j(\alpha) + \Sigma_i (s_i - \theta_i) u_i(\alpha)$$

この値は如何なる状態変化によっても増加することはない．この性質を利用して，適当に入力の重みを設定してやることにより，最適化問題に適用することができる．ユニットの状態遷移に確率を導入したものとして Boltzmann Machine がある．

逆伝播学習 (Back Propagation)

D.E.Rumelhartらによって考案された学習アルゴリズム．複雑なパターンの対応を学習

できる．各ユニットは［0,1］の連続量を状態として取る．ネットワークは入力層と出力層を含む3層以上の多層構造をなす．入出力以外のユニットを隠れユニットと呼ぶ．例題パターンの組を入力ユニットと出力ユニットに与えると，実際の出力と正解の出力パターンの間の誤差の2乗和が最小となるように各結合の重みを漸次，修正する．修正の式は次のとおりである．

$$\Delta w^{k-1}{}_i{}^k{}_j(t+1) = -\varepsilon \delta^k{}_j o^{k-1}{}_i + \alpha \Delta w^{k-1}{}_i{}^k{}_j(t)$$

$$\delta^m{}_j = (o^m{}_j - y_j) \cdot f'(i^m{}_j)$$

$$\delta^k{}_j = \sum_i w^k{}_j{}^{k-1}{}_i \delta^{k-1}{}_i \cdot f'(i^k{}_j)$$

$w^{k-1}{}_i{}^k{}_j$は入力層から数えて第k-1層のユニットiから第k層のユニットjへの入力の重みである．$i^k{}_j$は第k層のユニットjの入力，$o^k{}_j$は出力である．$y_j$は出力ユニットjが出力すべき正解の値である．出力層は第m層に当る．すなわち$o^m{}_j$は出力ユニットjの出力である．実際の出力と正解との間の誤差を3番目の式に従って出力側から入力側へ伝播させることによって重みを修正する．$\varepsilon$と$\alpha$は1回の修正の大きさを決めるパラメータで，正の定数である．fは入力から出力を決定する関数，すなわち$o^k{}_j = f(i^k{}_j)$であり，$f'$はその導関数である．

競合学習（Competitive Learning）

パターンの分類あるいは特徴検出の教師なし学習を行なう能力をもつ．各ユニットは状態として連続値をとり，多入力多出力である．ネットワークは一番下の入力層を含む多層構造をなす．隣接する層との間の結合は受け手側のユニットの状態を強化する（値を大きくする）ように働き，層の内部での結合は抑制する（値を小さくする）ように働く．初期状態では各結合の強さを適当なばらつきをもったランダムな値を割り付けておく．例題のパターン入力に対して強く活性化されるユニットについて，その入力アークの重みを増加させる．入力層以外の各層の内部では相互に抑制し合うユニットの集りが形成され，ある入力に対しては，その集りの中の1つのユニットが際立って強く活性化されるように学習が進められる．すなわち，各層における個々のユニットは，入力パターンに関するなんらかの特徴を発見する能力を持つことになる．

（2.1.5～2.1.7節担当　臧見）

## 2．2　知識処理技術

### 2.2.1 知識表現

知識をシステムが利用できる記号の列として記述するための形式であり，以下に示すものを含めて多様な形式がある．知識表現形式に求められる特性は，簡潔な表現であること，宣言的に記述できること，記述に柔軟性があること，各々の知識が独立して記述できると同時に，全体の関連性が把握できること，等がある．現在用いられている表現は主として浅い知識の記述に向いているが，深い知識の記述に適した表現はまだ実用化されていない．

キーワード：意味ネット，フレーム，ルール，オブジェクト指向表現，知識のコンパイル

意味ネット (semantic net)

個々の概念を表わすノードと，概念間の関係を表わすリンクとから構成されるネットワークのこと．例えば「リンゴの色は赤」という知識は意味ネットでは「リンゴ」というノードから「赤」というノードへ「色」という名前の（有向）リンクを引くことで表現される．リンクの持つ意味は，各概念の持つ属性・概念相互の階層関係（上位概念・下位概念）・全体―部分関係などがある．セマンティックネットともいう．

フレーム (frame)

具体的・抽象的な対象が持つ属性と属性値との集まりを表わすデータ構造のこと．例えば「リンゴ」フレームには「色＝赤」，「産地＝長野県」等の属性・属性値とを記述できる．また，通常はフレーム間の上下階層を表わすリンクを持ち，属性値の検索時にそのフレームに属性がなければ上位フレームからデータを継承（インヘリット）したり，データが検索・修正された場合に自動的に起動される手続き（デモン）を持つこともできる．

ルール (rule)

「IF …… THEN ……」の形式で記述された，判断部―帰結部表現のこと．前提に対する結論，条件に対する行動を表わす（「IF　熱が高い　THEN　風邪である」，「IF　風邪である　THEN　薬を飲む」）．一般に「IF A THEN B」は，「A ならば Bである（B をする

）」，「B である（B をする）ためには Aが必要である」という2種類の解釈ができ，前者は前向き推論，後者は後向き推論で用いられる．

オブジェクト指向表現 (object oriented expression)

具体的・抽象的な対象を，各々に固有なデータと手続きとで表現すること．表現された（広い意味での）データ構造をオブジェクトという．固有データはオブジェクトの内部状態を表わし，固有手続きはオブジェクトが実行する動作を表わす．固有手続きは複数持つことができ，これらを総称してメソッドという．複数のオブジェクトに共通なメソッドを代表して記述した上位オブジェクトをクラスオブジェクトといい，最下位層での，固有データをもつオブジェクトをインスタンスオブジェクトという．メソッドは上位オブジェクトから継承でき，またオブジェクトのメソッドを動作させる手段をメッセージ送信という．

知識のコンパイル (knowledge compile)

一般的・原理的・統一的な知識（深い知識）から具体的・個別的・特定的な知識（浅い知識）を生成することをいう．深い知識は適用範囲は広いが推論に時間がかり，一方浅い知識は推論を高速化できるが適用範囲が狭い．したがって広い適用範囲と高速な推論とを両立するために，少数の深い知識から多数の浅い知識を生成して用いる方法が考えられるが，これを自動化する知識コンパイラはまだ研究段階である．

### 2.2.2 推論機構

知識表現形式で記述された知識を入力データとして動作するインタプリタが推論機構である．動作の違いによって各種の推論方式を実行するインタプリタができるが，初期の，特定の推論方式のみ可能な単純なものから，現在では複数の推論方式をサポートするために複雑化し，少なくとも前向き推論と後向き推論の両方を行なえるものが一般的である．このため高機能で高度な推論ができるが，一方で推論機構の制御としてのメタ知識が必要になることがある．

キーワード：演繹推論，プロダクションシステム，前向き推論，後向き推論，事象駆動推論，予測駆動推論，黒板モデル

演繹推論 (deduction reasoning)

一般則としての推論規則と特殊例としての事実とから，新たな特殊例としての結論を導く方法を演繹推論という．「全ての人間は死ぬ」（一般則）と「ソクラテスは人間である」（事実）とから「ソクラテスは死ぬ」（結論）を導くのがその例である．したがって導出された結論は一般則以上の情報を持つものではないが，システムとして実現が容易なため現状の推論機能の主流となっている．

プロダクションシステム (production system)

ルールの集合・推論機能・ワーキングメモリから構成され，ルールの前提部でワーキングメモリを参照し，帰結部でワーキングメモリを更新する操作を行うシステムをプロダクションシステムという．ルールの集合をルールベースあるいは（狭義の）知識ベースという．現在は純粋なプロダクションシステムは少なく，フレーム等を追加したり，手続き（通常のプログラム）も扱えるものが多い．

前向き推論 (forward reasoning)

ルールを前提部→帰結部の順に適用し，事実として与えられたデータから導出される結論を求める方法を前向き推論という．１回のルール適用での帰結部での結果を中間結論という．現在ワーキングメモリ内に存在するデータ（中間結論の集合）を用いて新たなデータを生成しつつ推論する方法といえる．一般には，最初の解へ到達するまでに余分な中間結論が生成される欠点がある．データ駆動型推論ともいう．

後向き推論 (backward reasoning)

解として想定される目標（仮説）を設定し，ルールを帰結部→前提部の順に適用して仮説を検証する推論方法を後向き推論という．このときルールの前提が仮説になり，ワーキングメモリに同等のデータが存在するとき仮説が検証されたとみなす．データが存在しなければ現在の仮説を帰結部に持つルールを適用する．想定される解が多数あると各々の解について推論を実行するため時間がかかる欠点がある．目標駆動型推論ともいう．

事象駆動型推論 (event driven reasoning)

ある事象（ルールの帰結部を実行した等）の発生とその場合に行なう動作とを定義し，実行時に発生する事象に応じて，対応する動作を起動する方式をいう．一般には起動された動作によって新たな事象が発生するため，事象－動作の連鎖で推論が進行する．発生し得る事象と，対応して起動する動作とが事前に既知でなければならないが，木構造の探索は事象駆動によって容易に実現することができる．

予測駆動型推論 (expectation driven reasoning)

ある事象の発生において，次に実行される可能性のある動作とその動作の実行開始条件とを定義し，事象発生時には起動し得る動作の登録のみ行ない，実行開始条件が成立した時点で動作を起動する方式をいう．したがって登録時には各動作は実行待ちの状態にあるため，デモン機能として用いることができる．一方，デッドロックに陥る可能性もある（2つの動作の解除条件が，相手が起動されることであるような場合）．

黒板モデル (black board model)

複数の，知識源と呼ばれるルールベースと，黒板と呼ばれる分割されたワーキングメモリ，および制御機構から構成され，各知識源が黒板の特定の領域を参照・更新することにより全体として非同期的・協調的に動作する問題解決モデル．一般には完全な非同期ではなく，知識源を選択するスケジューラが制御機構内に存在し，知識源の起動順序をスケジュールすることで推論が制御される．制御方式が問題依存のときはメタ知識に対応する．

### 2.2.3 知識獲得

問題領域の知識をエキスパートシステムのソフトウェアとして実装（インプリメント）し，システムの問題解決能力を領域専門家と同等のレベルまで引き上げる過程が知識獲得である．一般に専門家の持つ知識は第三者に理解可能な言語・図表等の形式では事前に存在していないため，インプリメントに際しては言語化・視覚化を経てツールの表現形式で記述するというステップをとる．このようなステップとして，抽出・変換・構造化・洗練化があり，各々の意味を以下に述べる．

キーワード： 知識抽出，知識変換，知識構造化，知識洗練化

知識抽出 (knowledge elicitation )

エキスパートシステムにおける問題解決に必要な（主としてヒューリスティックな）情報（知識）を，第三者に理解可能な言語化・図表化して得る段階である．知識は，その問題領域に共通なもの（領域専門家なら誰でも用いているもの）と各専門家固有のものとがある．前者は文献・マニュアル等の文書から得られることが多いが，後者は一般に言語化・文書化しにくいため，専門家へのインタビュー，作業の観察，弟子入り等を通じて得る場合が多い．

知識変換 (knowledge translation)

抽出された知識をツールが提供する知識表現形式で記述する段階である．知識抽出段階での表現形式がそのままツールでの表現形式であることが理想であるが，現状ではルール・フレーム・オブジェクト等で記述することになる．専門家の判断・行動や，因果関係の連鎖はルールで記述し，問題対象（例えば診断の対象物）の構成要素・階層関係はフレームで表現されることが多い．

知識構造化 (knowledge structurization)

抽出された直後の知識は断片的なものであるため，専門家がそれらをどのように統合して用いているかを把握し，問題解決の流れを同定するために知識相互の関連性を明確化する段階．専門家は問題解決のために1)対象をどのように表現し（→対象の構造，解・仮説の表現形式），2)どのように使っているか（→部分問題への分割・階層化，仮説の設定・評価方式）等に着目し，問題のタイプに応じて決まるようなカテゴリに基づいて整理する．例えば診断問題では症状の検出，原因の推定，対策の指示等のカテゴリであり，計画問題では候補の生成，候補の評価等である．

知識洗練化 (kowledge refinement)

システムの問題解決能力の向上のために，知識ベースを整備・拡張する段階．構造化によって発見された，知識の不備な部分を補い，更に表層的な知識から深い知識を引き出し，あるいは複数の知識を統合してマクロ化する，等でシステムが対処可能な問題範囲を広げることを目的とする．また，メインテナンス性の観点から知識構造・表現の可読性を高め

ることも必要になる．

### 2.2.4 知識ベース管理

システムの問題解決能力が，知識量の増加に見合って向上しないことがある．知識相互の矛盾による誤った結果，他の知識に包含されるような無駄な知識の増加による実行速度の低下，推論制御のための知識の増加による全体動作の不透明化，等である．これらは高機能な推論機構によってある程度軽減されることもあるが，多くは人手による管理を必要とする．断片的な知識の全体を事前にどこまで有機的に整理・統合できるかが，知識ベースを管理し易くする鍵になる．

キーワード：矛盾の解消，冗長性の解消，知識ベースのコンパイル

矛盾の解消 (contradiction resolution)

ある入力データに対し，一つの結論とその否定とが推論される場合は矛盾が発生したことを意味する．この矛盾を解消する方法には，推論過程で仮説を用いている場合にはその仮説を変更する（ＴＭＳ），確信度のような値が付加されている場合には値の更新計算によって全体の整合性を回復する，等がある．これらはある程度自動化できるが，知識ベース自体が矛盾している場合には知識整理から見直す必要がある．

冗長性の解消 (redundancy resolution)

知識量が増加すると類似した知識が多くなる．このような場合には，重複内容の統合・他の表現への言い換え・上位概念の設定等により無駄を省き，知識構造の見通しをよくすることが重要である．これにより知識ベースの可読性が高まり，知識の不足・不備な部分を見出すことが容易になるため，知識ベースの拡張・修正等のメインテナンス性が向上する利点がある．

知識ベースのコンパイル (knowledge base compile)

知識ベース＋推論エンジンという構成においては，後者が前者のインタプリタとして動作するため，知識ベースのプロトタイピング性や可読性は良いが，完成後は実行速度やセ

キュリティの面で弱くなる．これらの欠点を改善するため知識ベースを例えばLispや Cのプログラムに変換し，さらに機械語に落とすことで実行速度とセキュリティとを向上させる方法が採られることがある．知識自体の性質は変化しないため，知識のコンパイルとは異なる．

（ 2.2節担当　中島）

# ３．知識工学技術

# 3. 知識工学技術

## 3.1 知識システムの構成

### 3.1.1 知識システムへの要請

知識システムは，数学のように明確な構造をもたないために，従来のアルゴリズム的な問題解決手法や手続き的プログラミング手法が適用できない「悪構造」な問題を解決するための手法である．

キーワード：悪構造問題，プロトタイピング的接近，システムの透明性，
ユーザーとの対話性，保守性・拡張性

悪構造問題(ill-structured problems)

「良構造問題」とは，数学のような明確な構造をもち，従来技術（アルゴリズム的な問題解決手法や手続き的なプログラミング手法）でモデル化や実装が可能な問題をいう．

一方，「悪構造問題」とは，明確な構造をもたず，従来技術（アルゴリズム的な問題解決手法や手続き的なプログラミング手法）でモデル化や実装が困難視されている問題をいう．知識システムは，後者の問題を対象とし，定性的あるいは断片的な知識を引き出してシステム化することが要求されている．

プロトタイピング的接近(system analysis by making prototype systems)

知識システムの構築に必要な知識の量と質を見極めるシステマティックな方法がないため，知識システムの構築には試行錯誤が必要である．また，領域専門家とＫＥとユーザーとは別の人間であり，領域専門家は特殊な知識をもつ人間であるため，三者を結ぶ共通の媒体が必要である．そこで，プロトタイプといわれる試作モデルをつくり，領域専門家とユーザーに見せ，システムに対するユーザーの要求仕様や問題解決の手法を事前に明確化する作業（専門家モデル及びユーザーモデルの同定等）が必要となる．

システムの透明性(readability of knowledge systems)

知識システムは，その内容が可読で，その挙動が分かるように構築し，システムの段階的開発及び構築後のメンテナンスの容易化を図らなくてはならない．

ユーザーとの対話性(communication with users)

ユーザー・インターフェイスを備え，推論の根拠や筋道を明確に示し，システムの内部的表現やメカニズムについて知らないエンドユーザーにもシステムとの対話を自然な形で行えるようにしておくべきである．必要に応じて，自然言語やグラフィックスを利用する必要がある．

保守性・拡張性 (maintenance and improvement of knowledge systems)

知識システムは，「悪構造問題」を対象としているため，完成ということは有り得ない．従って，適用結果をフィードバックし，知識の追加・修正を逐次行っていく必要がある．そのためには，知識システムの保守及び拡張がユーザー自身に容易に行える構造をとらねばならない．

### 3.1.2 知識システムの構造

知識システムは「推論機構」，「知識ベース」及び「ワーキングメモリ」とから構成され、「知識ベース」と「外界からの情報」及び「エンドユーザーから入力された情報」に基づき「推論機構」が推論を実行し，その度にワーキングメモリを更新していく．

キーワード: 知識ベース，ワーキングメモリ，推論機構

知識ベース(knowledge base)

知識ベースは，知識システムの主要部分である．知識を大別すると，宣言的知識と手続き的知識に大別される．知識表現には，「論理式」，「フレーム」，「プロダクションルール」，「意味ネットワーク」などがある．領域専門家の知識をその適切な表現方法に合わせて記述し計算機のメモリーに蓄積し，必要に応じて引き出せるようにしたものが知識ベースである．

ワーキングメモリ(working memory)

問題の初期条件の記述，推論によって得られた中間仮説及び結論，センサー情報及びエンドユーザーからの入力データなどが格納され，推論によって逐次更新される．

推論機構(inference engine)

与えられた問題に対して，知識ベースを利用して，問題解決を達成するための制御を行う機構である．

知識ベースと推論エンジンは，通常のプログラムにおけるデータ構造とアルゴリズムの関係に相当する．

3.1.3 知識システムの動作

知識システムは，各ルールの中から条件部の合うルールを選び出し，さらに選ばれたルールの中から，予め設定した評価基準に従って実行すべきルールを一つ選び，実行する．ルールの実行にともない，ワーキングメモリが更新される．

キーワード：パターンマッチング，競合解消，ワーキングメモリの更新

パターンマッチング(pattern matching)

プロダクションメモリにある各ルールの条件部とワーキングメモリの間でパターン照合を行い，条件を満たす全てのルールを見つけ出す操作をいう．この操作によって見い出されたルールは，互いに競合することになる．選ばれたルールの集合を「競合集合」と呼ぶ．

競合解消(conflict resolution)

先のパターンマッチングにより選ばれたルールの中から，予め設定した評価基準に従い特定のルールを一つ選択する操作をいう．

競合解消法として，「重要度優先方式」，「詳細ルール優先方式」，「最近実行ルール優先方式」などが挙げられる．

ワーキングメモリの更新(update of working memory)

マッチングにより選ばれたルールの結論部が要請している操作をワーキングメモリの内容に対して施すことにより内容を更新する.

(3.1　担当: 森谷)

## 3.2 構築方法論

### 3.2.1 知識システム開発手順

知識システムを効率良く構築するためには，構築手順とその作業内容が明かにされている必要がある．知識システムもソフトウェアシステムであり，システムアナリシス，プログラミング，保守運用といった基本的な構築手順には，従来と差がない．しかしながら，対象とする問題が悪構造を持った専門領域の問題であり，これらを効率良く取り扱うためプロトタイピングや知識プログラミングといった手法が，開発手順の中で用いられる．

キーワード：システムアナリシス，プロトタイピングアナリシス，知識プログラミング，保守・運用

システムアナリシス(system analysis)

ソフトウェア品質の良否は，ユーザの満足度である．したがって，知識システムにはどの様な機能が必要か，まずユーザの立場で分析する必要がある．知識システムは領域専門家の作業を代替しようとするものなので，主に専門家へのインタービュー，弟子入り，プロトタイピングといった方法通じ，その実現法である知識が得られる．また，最適なシステムを構築するため，知識システム技術と従来技術のトレードオフが，この段階で決定される必要がある．

プロトタイピングアナリシス(prototyping analysis)

知識システムが取り扱う問題は，悪構造をもったものが多い．したがって，的確なユーザ要求を反映したシステムの要求仕様を短期間にまとめる目的や，この要求の実現性をシステム開発の早期に知る目的のため，実用システム以前に実際に動作するプロトタイプを作成し，これらを評価する必要がある．このプロトタイプには，増殖的に実用システムに移行するものと，これらの目的のため使われ使い捨てされるものがある．

知識プログラミング(knowledge programming)

獲得された領域専門家の知識を，プログラムすることである．従来のプログラムでは，知識と問題解決機能（推論機構）が一体となっているため変更容易性を欠き，悪構造を有する問題向きでない．したがって，両者を分離し，知識のみをプログラムすることでシステム構築が可能なプログラミング技法が用いられる．知識プログラミングを容易にするために，知識の性質によってルール，フレーム等の各種知識表現技法を選択する．

保守・運用(maintenance)

知識システムは，ＫＥを中心としたシステム開発部門で開発され性能評価を行なった後，ユーザ部門へ導入される．また，知識システムは増殖的な（incremental）性質を有するので，ユーザ部門によるシステム拡張を可能にする必要がある．したがって，その後の保守運用にあたっては，知識ベースの管理等といった受け入れ体制作りも重要である．

### 3.2.2 問題の類型とタスク

従来の知識システムは，ルールやフレームといった実装レベルの知識表現法によって行なわれていたため，実現したいタスクと知識表現間の隔たりから，その構築が困難であった．したがって，知識システムを効率良く構築していくためには，出来るだけ対象とする問題有するタスクに近い知識表現法を採用していく必要がある．このような観点から，知識システムが取り扱う問題をその性質から分類し，その上で分類された問題が共通に有するタスクを挙げる．

キーワード：解析型問題，計画・設計型問題，類型的タスク

解析型問題(analysis type problem)

知識システムが取り扱う問題は，その問題解決の対象となっているシステムの特性によって解析型問題と設計・計画型問題へ大別出来る．前者の解析型問題は，問題解決の対象となっているシステムにおいて，そのシステムの構造とサブシステムの特性が与えられているとき，システムの特性を推測するような問題である．これらの型の問題は，更に解釈問題，診断問題，制御問題に分類される．

計画・設計型問題(scheduling and design type problem)

知識システムが取り扱う問題は，その問題解決の対象となっているシステムの特性によって解析型問題と計画・設計型問題へ大別出来る．後者の計画・設計型問題は，問題解決の対象となっているシステムにおいて，そのシステムの特性が与えられたとき，これを実現するようなシステムの構造とサブシステムの特性を決定するような問題である．この型の問題は，更に計画問題と設計問題に分類される．

類型的タスク(generic task)

Chandrasekaranらは，従来の知識システムがルールやフレームといったインプリメンテーションレベルの知識表現モデルに基づき構築されていることで，計画や診断といった複雑なタスクの表現を困難にしているとし，これらのタスクに共通する知識処理の型（類型的タスク）を抽出し，これらの類型タスクの組み合せで知識システムを構築することを提唱した．この様に，個々の問題にあまり強く依存しない汎化されたタスクではあるが，これら結合によって，知識システムが構築可能なタスクのことを言う．

### 3.2.3 解析型知識システムの構築

知識システムが取り扱う問題は，その問題解決の対象となっているシステムの特性によって解析型問題と計画・設計型問題へ大別出来る．前者の解析型問題は，問題解決の対象となっているシステムにおいて，そのシステムの構造とサブシステムの特性が与えられているとき，システムの特性を推測するような問題である．この型の問題は，更に解釈問題，診断問題，制御問題に分類される．これらの問題の具体的な特徴と知識システム化の際のアプローチ法について述べる．

キーワード：解釈問題の特徴，診断問題の特徴，制御問題の特徴

解釈問題の特徴(characteristic of interpretation problem)

計測データの特徴を抽出し，その物理的意味を解釈する問題である．その特徴として，データ間の高次相関，雑音の混入，誤りの存在，既存信号処理との統合，対象システムのモデル，感覚データの認知モデル等の問題がある．知識システムへの接近の基本は，対象システムの機能や構造の知識（深いモデル）による推論と経験的知識（浅いモデル）による推論の統合にある．

診断問題の特徴(characteristic of diagnosis problem)

異常や故障の発生時，対象システムの知識と観測データから原因箇所を同定する問題である．特徴として，設計及び経験的知識や操業データの利用，計測時間やコスト，診断の対話性，知識の不確実性や抽象化等の問題がある．知識システムへの接近は，経験的知識，設計知識，類型タスクによるものがあり，問題にあった手法を選択することが重要である．

制御問題の特徴(characteristic of control problem)

システムの状態を監視し，予定された状態になるようにシステムを制御することである．特徴として，数学的なモデルが立たずかつ時定数が大きく制御理論の適用困難，実時間性や信頼性の制約が大きい，既存システムとのインターフェースが多い等といった問題がある．知識システムへの接近は，高信頼要求部は制御理論による従来法を，大局的方針決定部は知識システム化を，というように信頼性の面から両者の融合を図ることが重要である．

### 3.2.4 計画・設計型知識システムの構築

知識システムが取り扱う問題は，その問題解決の対象となっているシステムの特性によって解析型問題と計画・設計型問題へ大別出来る．後者の計画・設計型問題は，問題解決の対象となっているシステムにおいて，そのシステムの特性が与えられたとき，これを実現するようなシステムの構造とサブシステムの特性を決定するような問題である．この型の問題は，更に計画問題と設計問題に分類される．これらの問題の具体的な特徴と知識システム化の際のアプローチ法について述べる．

キーワード：計画問題の特徴，設計問題の特徴

計画問題の特徴(characteristic of scheduling problem)

目的を達成するため，利用可能な資源（人，物，金）の割当を時系列として決定することである．特徴としては，広い探索空間，多様な評価属性，不確実な環境予測，部分計画間の相互作用等といった問題がある．知識システムへの接近は，問題のタイプによって状態空間探索，部分問題への分割，数理計画法の適用等がある．

設計問題の特徴(characteristic of design problem)

与えられた要求仕様を満足するシステムの構造とその各構成要素の仕様を決定することである．特徴としては，広い探索空間，解析による合成，構成要素の最適化，検証の完全性と効率性等といった問題がある．知識システムへの接近は，仮のシステムを生成評価しその改善を繰り返す解析による合成（生成検査法）と，システムをマクロな表現から段階的に詳細化するトップダウン精密化が基本的である．

（3.2節担当　菊地）

## 3.3 知識獲得支援

### 3.3.1 開発段階と知識獲得

知識システムの開発作業は知識獲得の面からいくつかの段階に分けて考えることができる．最近では，システム分析に相当する初期の段階から知識の評価にいたる開発の広い範囲を知識獲得としてとらえる考え方もあるが，ここでは知識獲得の基本問題として知識の抽出，変換，構造化，洗練化および知識ベース管理をとりあげる．以下，各段階で行われる作業，関連する技術動向，留意点などについて述べる．

キーワード：知識抽出，知識変換，知識構造化，知識洗練化，知識ベース管理

知識抽出(knowledge elicitation )

設定された問題の解決に必要な知識を専門家から収集し，はっきりした記号（言語）で表現する段階．この際，知識の網羅性を確保することが重要であるが，人間の記憶が連想型であることに留意する必要がある．なお，専門家から抽出される知識は一般に理論または経験に裏うちされてはいるものの，コンパイルされた手続き的な浅い知識が多い事に注意する必要がある．

知識変換(knowledge transfer)

抽出された知識を管理・維持・活用するために，適切な枠組みによって形式的に表現する段階．知識システム構築支援ツールを用いる場合には，知識システム技術者がそのターゲット言語に変換する．また，対象領域に関するモデルで表現すれば，専門家自身がシステムと対話的に知識の投入作業を行うことができ，効率よく信頼性の高い知識ベースが開発できる．

知識構造化(knowledge structurization)

変換された知識を，知識相互の関連性によって明確な形に整理したり，知識が持つ階層性などの構造を見出すことにより，問題解決に有効に利用できるようにする段階．システム工学などで従来使われてきた，構造モデル化技法やＫＪ法などを援用する研究がなされ

ている．また，様々な情報の関連付けを操作管理するハイパーテキストの利用も今後重要になると考えられる．

知識洗練化(knowledge refinement)

知識ベースの静的・動的な解析により新しい知識を得て知識ベースに取り込む段階．例えばテストケースについての専門家の評価により知識ベース内の弱点を見出したり，問題解決過程から知識のチャンキングを行って性能を向上させることなどがこれにあたる．専門家との対話により洗練化を支援するシステムとしてはＳＥＥＫやＭＯＬＥなどがある．また，複数の問題解決例を使って統計的分析に基づいて自動的に洗練化を行うものとしてＳＥＥＫ２が開発されている．

知識ベース管理(knowlege base management)

知識ベースの無矛盾性および冗長性を管理するとともに，問題解決に必要な知識がすべて網羅されているかどうかを検証する段階．理論的な検証方法の確立およびシミュレーションによる検証技術が要請される．仮説推論，ＴＭＳ，ＡＴＭＳなどの技術が重要な役割を果たす．また，ＣＹＣの知識ベース開発用エディタは大規模な知識ベースの管理に有効な支援機能を実現している．

### 3.3.2 知識獲得支援技法

専門家からの知識の獲得は，通常，知識システム技術者との共同作業によって行われる．この作業をスムーズに進める能力は，知識システム技術者に要請される重要な素養のひとつである．ここでは，代表的な技法としてインタビュー技術とプロトコル解析とをとりあげる．なお，知識獲得を組織的に行うためにはこれらの技法に加えて種々のフェーズに対応した支援技術が必要となる事に注意されたい．

キーワード：インタビュー技術，プロトコル解析

インタビュー技術(interview)

知識システム技術者が専門家との対話を通して知識を獲得するための技術．現状のエキ

スパートシステムの開発では大半がこの方法を用いている．知識システム技術者には，知識システム開発技術に加えて，対象領域の基本的な知識，専門家の心理的特性の理解など様々なものが要請される．また，インタビュー戦略を体系化し知識獲得支援システムに応用する研究も行われている．

プロトコル解析(protocol analysis)

専門家に実際に問題を解いてもらい，その活動を観察することによって，推論戦略や用いられた知識などを解析する手法．意思決定の過程で考えていることを声に出して説明してもらう「発語思考法」がよく用いられる．この手法は，専門家が自分の知識を適切に表現しにくい場合に有効である．しかし，現実に使われている知識を導きだすためには注意深い検討が必要である．

### 3.3.3 知識獲得支援ツール

知識獲得支援ツールは，知識獲得の作業をできるだけ機械で支援することをねらったソフトウエアシステムであり，現在，実用的な面からも理論的な面からも，積極的な研究開発がなされている．初期のシステムとしては知識ベースの保守過程を支援するＭｅｔａ－ＤＥＮＤＲＡＬやＴＥＩＲＥＳＩＡＳなどが知られている．ここでは最近のツールを次のキーワードに従って分類し紹介する．

キーワード：発想支援型ツール，連想支援型ツール，類型的タスクに基づくツール，領域モデルに基づくツール，汎用指向ツール

発想支援型ツール(conception support tool)

ブレーンストーミング，ＫＪ法，などの発想を支援する手法を機械化することにより，知識システム開発の初期段階を支援することを目的とするツール．このようなシステムの例としては，ゼロックスＰＡＲＣのコンピュータ黒板Ｃｏｌａｂや，豊橋技術科学大学のＫＪエディタなどがあげられる．

連想支援型ツール(association support tool)

専門家に専門作業のタスクと関連知識の連想を促すことにより，スムーズな知識獲得を行うことを目的とするツール．人間の記憶が連想型であることを利用している．ＩＣＯＴで開発されたＥＰＳＩＬＯＮ／ＥＭで用いられているプリ・ポスト法では，あるオペレーションに関係する前後のオペレーションの連想を専門家に促す．

類型的タスクに基づくツール(generic-task-based tool)

知識システム開発における組織的アプローチを目指す，類型的タスク理論に基づくツール．知識の形式が対象問題に適した構造を持っているため，汎用のツールと比較して知識獲得がより自然に行われる利点がある．階層的分類を行うＣＳＲＬやＣＴＡＳ，および，類似設計を行うＤＳＰＬが知られている．また，富士通のＥＳ／ＳＤＥＭに基づく知識獲得支援ツール「ＳＡＫＡＳ」でも同様の概念が使われている．

領域モデルに基づくツール(domain-model-based tool)

対象領域の問題構造を分析して得られる概念モデルを利用して，領域専門家が知識システム技術者の援助なしに効率よく知識ベースを開発するためのツール．例えば診断型問題を対象とするＭＯＲＥでは原因仮説と徴候をノード，それらの因果関係をリンクとするネットワークで表現し，その部分構造を解析することにより初期知識ベースの洗練などの機能を実現している．

汎用指向ツール(domain independent tool)

知識獲得の初期段階において知識の抽出・整理などを行う，領域に依存しないツール．例えばＣＯＮＳＩＳＴは，ＫＪ法に基づいてフレームやルールを最終結果として考慮した知識整理を支援するシステムである．また，ハイパーテキストや，Ｃｏｌａｂ，ハイブリッドシェルの知識ベースエディタなども汎用の知識獲得支援ツールとみることができる．

(3.3節担当 山崎)

## 3.4 知識プログラミング

### 3.4.1 プログラミング言語

知識システムは専門家のもつ知識を記号で表現し，問題解決過程をモデル化した後，計算機上にインプリメントすることによって行われる．したがって知識システムの開発には記号処理に適する言語を利用したほうが，その開発は容易である．さらに知識システムは専門家モデルの同定やユーザモデルの同定のためプロトタイピング的に開発されることも多く，そのためには柔軟で強力なソフトウェア開発支援環境が必要とされる．本節ではそのような知識システム開発用としてよく用いられているプログラミング言語の特徴とその動向について説明する．

キーワード：　ＬＩＳＰ，ＰＲＯＬＯＧ，オブジェクト指向言語，Ｃ言語，
マルチパラダイム言語

ＬＩＳＰ

人工知能研究において最も多くつかわれており，開発環境も整備されている．知識システム開発ツール自体もＬＩＳＰで記述されているものが多く，そのようなツールを利用して知識システムを開発する場合ＬＩＳＰの知識は必須である．ＬＩＳＰには方言が数多くあるが，最近ではＣｏｍｍｏｎ　ＬＩＳＰが共通の言語仕様となっている．関数型言語としての特徴，リスト処理，変数束縛，再帰処理等を理解する必要がある．

ＰＲＯＬＯＧ

述語論理を制限したホーン論理に基づいたプログラミング言語であり，人工知能システムの研究，開発に最近よくもちいられている．プログラムの実行制御方法が従来型言語と大きく異なっており，ユニフィケーションとバックトラックを基本として実行制御を行う．学習にあたってはその特徴をよく理解することが必要である．

オブジェクト指向言語(Object-oriented Language)

手続きをオブジェクトのメソッドとして記述し，オブジェクト間のメッセージパッシン

グによりプログラムの制御を行うことに特徴がある．その結果プログラムのモジュール性が向上するので，知識システムの特徴であるプロトタイピング的開発が比較的容易となる．代表的なものにＳｍａｌｌｔａｌｋがあるが，ＬＩＳＰやＣ言語へのオブジェクト指向の導入も行われている．

Ｃ言語(C Language)

プロトタイプとしての知識システム開発にはほとんど用いられないが，問題解決手順や問題の構造が明確になった後の実システムの開発，制御など推論の高速性が要求される問題や，知識システムと既存システムとの接続のためのインタフェース言語として，知識システムの実用化段階では重要である．

マルチパラダイム言語(Multi-paradigm Language)

関数型，論理型，オブジェクト指向等，複数のプログラミングパラダイムを融合してひとつの言語として提供しており，問題に応じて適当なパラダイムを使うことができる．実際のシステムとしては，ＬＩＳＰをベースにオブジェクトや論理の取扱いを可能にしたものが多い．代表的なものとしてＬＯＯＰＳ，ＴＡＯがある．

### 3.4.2　知識システム開発ツール

知識システムの開発にあたってはＬＩＳＰ，ＰＲＯＬＯＧなど上述のプログラミング言語を用いるよりは，知識システム開発ツールを用いたほうが開発の効率は向上する．開発ツールは，以前の単一の知識表現や推論制御をサポートしたツールから，複数の知識表現や推論制御を可能とし様々な機能を実装したツールが提供されている．しかしながらそのような大規模汎用ツールは，プロトタイピングには適するものの，実用システムとして運用するには実行速度を始めとして問題となることもあるので，ツールの選択にあたっては考慮しなければならない．そのような汎用ツールに対して，最近では対象とする問題のタスクや領域に特化した専用ツールが開発されつつある．

キーワード：　ルール型ツール，ハイブリッド型ツール，タスクベースツール，
問題領域専用ツール

ルール型ツール(Rule Based Tool)

ＩＦ－ＴＨＥＮ型式で知識を表現しており，知識のモジュール化が可能である．その結果システムの保守拡張性に比較的すぐれている．しかしルール数が多くなると制御のための知識も多く必要となり，システムの透明性が低下するという知識の飽和の問題を持つ．知識システム開発ツールとしては最も基本的であり，知識システムの開発にはよく用いられている．代表的なものにＯＰＳ５がある．

ハイブリッド型ツール(Hybrid Type Tool)

ルール型ツールとフレーム型ツールの両方の機能を提供しているツールである．基本的な知識表現としてルールとフレーム表現が可能である．推論制御も前向き及び後向き推論が可能であるものが多い．さらにオブジェクト指向を導入するなどマルチパラダイム化している．多機能であるので扱える問題のクラスは大きくなるが，機能を使いこなすには訓練が必要である．代表的なツールにＫＥＥ，ＡＲＴ，ＫＣ等がある．

タスクベースツール(Task Based Tool)

これまでのツールは問題の種類や領域によらない汎用ツールであるのにたいし，診断問題用ツール，スケジュールング問題用ツールなど，特定のクラスの問題に特化したツールをタスクベースツールという．汎用ツールはプログラミングレベルを支援するのにすぎないが，タスクベースツールではモデリングレベルでの支援を目的としている．代表的なツールとして類型的タスク理論に基づく，分類・診断問題用のＣＳＲＬやルーチン設計問題用のＤＳＰＬがある．

問題領域専用ツール(Domain Specific Tool)

化学プロセス設計用ツール，プロセス制御用ツールなど対象とする問題領域に特化したツールのことを言う．対象とする問題領域に共通な知識をツール自体がすでにもっていることが特徴である．優れたグラフィックインタフェースや，シミュレーション機能をもっているものも多い．市販されているシステムとしてＧ２，ＰＩＣＯＮ，ＩＮ－ＡＴＥなどがある．

### 3.4.3 従来技術との接続

プロトタイプ段階は別として，知識システムを実用化する段階では，知識システム単独で運用される場合よりも，既存のソフトウェアシステムの一部としてあるいは既存システムと情報の交換をしながら運用される場合が多い．したがって知識システムの開発にあたっては，知識システムとして開発する境界の見極めや，データベースやシミュレータなどの既存のソフトウェアシステムとの接続，またセンサーや計測制御機器など既存のハードウェアシステムとの接続な従来技術との接続に関して，十分に考慮しながら開発を進めなければならない．

キーワード：　データベースとの接続，既存ソフトウェアとの接続，
計測制御機器との接続，シミュレータとの接続，実時間処理

データベースとの接続(Linkage to Database)

実用レベルの知識システムは，膨大な情報をもつデータベースから必要なデータを取り込み推論を行った後，その結果をまたデータベースに戻すなど，データベースと有機的に接続することが必要である．知識システム開発ツールの中にはデータベース管理システム（主として関係データベース）とのインタフェースを持っているものもある．関係データベースは述語論理と等価であることから，述語論理によってデータベースを体系化しようとする演繹的データベースなど，知識ベースとデータベースの関係は密接なものとなっている．

既存ソフトウェアとの接続(Linkage to Conventional Software System)

知識システムだけが単独で実用化されることは少なく，通常は既に開発されている従来のソフトウェアと接続してもちいられる．既存システムとの接続は，バッチ処理的なもので十分なのか，あるいはリアルタイム性が要求されているのかによって，接続方法は異なる．接続の方法としてはマシン間通信による方法，プロセス間通信による方法，またプログラム相互の埋め込みによる方法などが考えられる．既存ソフトウェアとの接続にあたってはこれらを考慮することが必要である．

計測制御機器との接続(Linkage to Sensing and Controle Device)

制御型の知識システムでは，対象システムからセンサーや計測機器を通じてデータを取り込み推論に用いる必要がる．計測機器からのデータの取り込みは高速性が要求されることも多い．したがって知識システムとのインタフェースには十分な配慮がなされなければならない．

シミュレータとの接続(Linkage to Simulater)

設計型知識システムに代表されるように，推論結果をシミュレータにわたして評価を行い知識システムにフィードバックするという機能が必要とされることもある．

実時間処理(Real Time Processing)

実時間処理に要求される機能としては，応答の高速性はもちろんのこととして割り込み処理や非同期処理などがあげられる．知識システムがガベージコレクションをおこす可能性がある場合は注意しなければならない．特に制御型知識システムではこれらの実時間処理に対する配慮が必要とされる．

### 3.4.4 ユーザインタフェース

ユーザインタフェースの充実は，知識システムに限らずどのようなシステムにおいても重要である．特に専門家の複雑な問題解決を代替，支援する知識システムにおいては，単に与えられた問題を解決するだけではなく，その問題解決過程を説明するなどエンドユーザがシステムの解の導出過程の妥当性を評価できるような機能が必要とされる．またシステムの入力は，冗長性を排除することなどは当然として，ダイレクトマニピュレーションによるなど人間工学的な配慮がなされなければならない．システムからのグラフィック機能を利用し結果を動的に視覚化して表現するなどユーザの直感的理解を容易にするように配慮すべきである．

キーワード:　マルチウィンドウシステム，ダイレクトマニピュレーション，
グラフィックス，自然言語処理，説明機能

マルチウィンドウシステム(Multi-window System)

マルチウィンドウシステムは高解像度のビットマップディスプレイとマウスに代表されるポインティングデバイスによって構成される．ユーザは同一のディスプレイ装置上で複数の独立した作業をそれぞれ異なる表示領域（ウィンドウ）に割り当てることにより，例えばシステムの実行テストを監視しながらエディットを同時に行うなど，効率的にシステムの開発を行うことが可能となる．通常マルチウィンドウシステムには下記のダイレクトマニピュレーション機能が実装されている．

ダイレクトマニピュレーション(Direct manipulation)

システムの操作を，システムに固有のシンタックスに基づいたコマンド言語によって行うのではなく，操作対象をアイコン表示しておきマウスなどのポインティングデバイスを用いて直接対象を直接操作したり，表示メニューを選択することでシステムの操作を簡単にすばやく行うことは，ユーザフレンドリーなインタフェースの具体例のひとつである．計算機になじみのないエンドユーザであっても迷わずに操作できるように人間工学的な配慮が必要である．

グラフィックス(Graphics)

エンドユーザがシステムの推論結果や入力に関する情報を即座にまた直感的に理解するには，単にテキストや数字の羅列でシステムからの出力を表示するのではなく，グラフ化やアニメ化などグラフィック機能を利用することにより，情報の視覚化，アナログ化を行うことがきわめて重要である．またシステムの開発者にとっては，よく用いられるようなグラフィックの部品は知識システム開発ツール側でサポートされていたり，開発時に推論の過程がグラフィックで示される等の開発支援環境が整備されていることが望ましい．

自然言語処理(Natural Language Processing)

日本語の本格的な自然言語処理は，現状の人工知能技術では達成することはできないが，簡単な自然言語解析であってもユーザインタフェースとして利用することは有用である．一般に自然言語処理技術と言われている中で，分かち書き処理行う形態素解析技術や，係受け処理を行う構文解析については技術として確立しつつある．知識システムへの利用としてはエンドユーザへの自然言語フロントエンドとしての利用が考えられる．

説明機能(Explanation Function)

知識システムが対象とする分野では，推論の結果だけでなく，その過程や推論の正当性の保証が重要であることが多い．したがって説明機能として，なぜシステムがユーザにその質問をする必要があるのかを示すＷＨＹ機能，推論の過程や結論にいたった理由を示すＨＯＷ機能，さらにシステムの解答がユーザの予想と異なっていたときになぜユーザが予想する解にならないのかをシステムにたずねるＷＨＹ　ＮＯＴ機能などが必要とされる．

（3.4節　担当　関根）

## 3.5 プロトタイピング

### 3.5.1 ソフトウェア開発のライフサイクル

ソフトウェアのライフサイクルは，対象問題の認識からはじまって製作，受入れ，保守にいたるまでの一連の作業，工程からなる．

従来型の大規模ソフトウェアシステムでは，通常，ウォータフォール型の開発手法がとられる．この手法では，開発の工程を問題認識，要求定義，設計，製作，受入れ，運用保守の6作業フェーズに分割し，各フェーズごとに作業が完結するようにトップダウンに作業を詳細化する．この手法によると開発の初期段階でユーザの要求を確定するので後の段階で修正が生じにくい．反面，ユーザの要求の変化に対応しにくくなる．

知識システムの開発には，一般に，プロトタイピング手法がとられる．これは，開発の初期の段階から，模型（プロトタイプ）のシステムを作成し，その評価をつうじてユーザの要求を明確化していく手法である．

キーワード：問題認識，要求定義，設計，製作，受入れ，運用保守

問題認識(Problem Inception)

ユーザの漠然とした問題解決への期待を明確に認識し，それをソフトウェア開発担当者に伝達するまでの段階である．

要求定義(Requirement Definition)

ユーザの問題認識に基づいて対象問題を分析し，それを解決する（機械的，人的な）システムの機能を定義する段階である．この結果はシステムの機能仕様にまとめられる．

設計(Design)

機能仕様の定めるシステムをソフトウェアとしてどう実現するかを決定する段階である．ここで静的なシステムの記述が，どう動作するかという実現方法の記述へと変換され，詳細化される．この結果は設計仕様にまとめられる．

製作(Implementation)

設計仕様にしたがって，特定のプログラム言語を使ってプログラムを製作する段階である．ここで，対象問題はコンピュータが理解できる形式に変換され，ソフトウェアとして機能するようになる．

受入れ(Acceptance)

製作したシステムが，ユーザの問題解決に役立つかどうかを検査し，システムを実際の運用に供する段階である．

運用保守(Maintenance)

運用されているシステムを保守する作業である．ソフトウェアシステムには，ハードウェアのような摩耗や劣化はない．ソフトウェアに保守が必要となるのは，製作したシステムに対象問題を解決する上でなんらかの誤りが存在した場合，ならびに，ユーザが解決したい対象問題の性質が変化した場合である．したがって，ソフトウェアシステムを運用している限り，それに対する保守作業は必ず必要となる．

### 3.5.2 プロトタイピングの型

知識システム開発に適するプロトタイピング手法には，その目的に応じていくつかの型が存在する．以下では，それをシステムのユーザ，開発者，ソフトウェア，ハードウェアの４つの視点の組合せで分類して示す．

キーワード：探査型，実験型，性能評価型，ハード評価型，インタフェース評価型，
機能評価型

探査型(Exploratory Prototyping)

ユーザと開発者の関係おいて，システム要求の抽出と評価とを目的として行われるプロトタイプ開発である．この場合には，作業の効率化に，高機能のプログラミング環境や柔軟なＡＩ言語の存在が有用である．

実験型(Experimental Prototyping)

開発者とソフトウェアの関係において，設計案のテスト・評価を目的として行われるプロトタイプ開発である．これはＡＩ研究者が古くからとっていた研究開発のアプローチと同様の喪のである．

性能評価型(Performance Investigating Prototyping)

ソフトウェアとハードウェアとの関係において，システムの実行性能を評価することを目的するプロトタイプ開発である．これによって，適切なハードウェア資源を決定したり，システム記述言語を変更したりする．

ハード評価型(Hardware Investigating Prototype)

開発者とハードウェアとの関係において，ハードウェアの適合性を評価することを目的としたプロトタイプ開発である．これによってシステム開発・保守作業に適したハードウェア資源を決定する．

インタフェース評価型(User Interface Investigating Prototype)

ユーザとハードウェアとの関係において，インタフェースの適合性を評価することを目的としたプロトタイプ開発である．これによって，システムの使いやすさを評価・検証することになる．

機能評価型(Function Investigating Prototype)

ユーザとソフトウェアとの関係において，システム機能の評価を目的としたプロトタイプ開発である．従来型システムの開発に際して用いられるプロトタイピング手法は機能評価の目的で利用されることが多い．

### 3.5.3 プロトタイピングのサイクル

従来のウォーターフォール型のソフトウェア開発では，システムの評価を受入れ段階で行う．それに対し，プロトタイピング手法によるソフトウェア開発では，開発初期の要求定義段階から，実際に動くシステムをユーザに提示する．これによってユーザの要求を明

確化しようとする．このプロトタイピングのサイクルを何回か繰り返しすことでシステムを徐々に成長させ実運用をめざす．このサイクルは，要求定義，プロトタイプの設計と利用，ユーザ評価，評価結果分析，プロトタイプの修正の５段階に区分することができる．

キーワード：要求定義，プロトタイプの実現と利用，ユーザ評価，評価結果分析，
プロトタイプの修正

要求定義(Requirement Definition)

ユーザの対象問題に対する認識が明確になった時点から，実際にソフトウェアとして稼働しうる部分の要求を順次記述する．

プロトタイプの実現と利用(Implementation by Prototyping)

開発したいシステムのうち，要求が明確になった部分をできるだけ速く実現し，以前のサイクルで開発した部分とあわせて，ユーザに利用させる．

ユーザ評価(Investigation with Users)

プロトタイプで実現した部分を実際に使用してその有用性を評価する．評価は，システムの機能，ハードウェア／ソフトウェア性能，ユーザインタフェースなどの視点から実施する．

評価結果分析(Analysis of the Results)

開発者が，ユーザの評価結果を分析し，プロトタイプとして修正すべき点を明確化する．そして，最終的なシステムの仕様を明らかにしていく．

プロトタイプの修正(Modifying the Prototype)

ユーザの評価が改善されるように，プロトタイプの修正を行う．ここでは，それ以前に開発したプログラムを廃棄することもある．

### 3.5.4 知識システムのプロトタイピングによる開発

知識システムは，悪構造問題を対象とする点，ユーザとの対話性を重要視する点，保守性・拡張性の向上をめざす点などから，プロトタイピングによる開発が適したシステムである．ただし，知識システムのプロトタイピングは，ＡＩ技術をベースとする点，知識処理が中心である点などから，従来型のシステム開発におけるプロトタイピングと異なる部分が多い．要求定義，プロトタイピングによる開発，専門家とユーザの評価，評価結果分析，プロトタイプの段階的発展の面から知識システム開発の特徴をまとめる．１章で示した知識システムの開発手順では，システムアナリシスの段階が要求定義に，プロトタイピングアナリシスと知識プログラミングの段階がプロトタイピングによる開発以後のフェーズに対応する。

キーワード：要求定義（専門家モデル，ユーザモデル），プロトタイピングによる開発
専門家とユーザの評価，評価結果分析，プロトタイプの段階的発展

要求定義(Requirement Definition)

知識システムの要求定義においては，専門家モデルとユーザモデルを明確化する必要がある．まず問題解決にたずさわる専門家の知識を専門家モデルとして構成するには，問題解決過程の分析，問題解決戦略の定式化，基本的な解集合の同定，対象モデルの記述，仮説集合の同定などの作業が必要となる．次に，ユーザモデルを構成するには，ユーザのもつ問題解決知識のレベル，利用時の主導権，ユーザインタフェース，ユーザによるシステムの保守・拡張の可能性などを明確にする作業が必要となる．

プロトタイピングによる開発(Implementation by Prototyping)

プロトタイピングによる開発を行う．この際，推論機構と知識ベースを特徴とする知識システムのアーキテクチャ，ならびに，ＡＩ技術による高度なプログラミング環境を適切に利用することによって，ソフトウェア開発の生産性を著しく改善することができる．

専門家とユーザの評価(Investigation by Users and Human Experts)

開発した知識システムは，システムの問題解決能力を確認するという点からは専門家にによって，システムの使い勝手や問題解決機能を確認するという点からはユーザによって評価されなければならない．

評価結果分析(Analysis of the Results)

専門家とユーザによる評価結果を分析して，要求定義段階で定式化した専門家モデルとユーザモデルの適切さを確認する．

プロトタイプの段階的発展(Evolution of the Prototype)

以上のプロセスを何回か繰り返し，プロトタイプを段階的に発展させる．通常は，推論機構の機能確定，知識ベースの拡張，ユーザインタフェースの充実という順序で，プロトタイプシステムの段階的な発展が行われる．

（3.5節担当　寺野）

## 3.6 知識システムの開発管理・運用

### 3.6.1 知識システムの開発管理

知識システムの開発に際しては，まず見積りを行う必要がある．見積りには，その対象によってコスト見積り，規模見積り，期間見積りがある．知識システムは，システム設計時に知識の総量が明確になっていない場合が多く，知識ベースの規模の正確な推定が難しい，またその規模が漸進的に拡張されるという特徴を持つ．従って，見積りの際にはこの点を十分に考慮しなければならない．

また，知識システム開発における管理項目としては，工程管理，品質管理，原価管理，製品管理の４つがあげられる．

キーワード: コスト見積り，規模見積り，期間見積り，
工程管理，品質管理，原価管理，製品管理

コスト見積り (cost estimation)

従来システムでは，経験に基づいて，規模，平均生産性，人数をベースにコストを見積っている．知識システムの場合，知識ベースの質が知識システムの完成度に密接に関係しており，知識ベース開発のコストの見積りが重要である．

規模見積り (program scale estimation)

従来，ソフトウェアの規模は，プロトタイプや仕様書をもとに推定されており，通常，ソースコードの行数等が用いられている．知識システムの場合，知識の総量が明確でなく，また，知識表現形式も，ルール，フレーム等各種あるため，知識ベースの規模の正確な推定方法の検討が必要である．

期間見積り (development time estimation)

一般に，ソフトウェアの開発期間は，その規模を平均生産性と人数との積で割ることによって得られる．知識システムにもこの算式を適用できるが，知識ベースの平均生産性については，単に量的問題だけでなく，知識獲得の難易度，知識が体系化され

ているかどうか，矛盾した知識はないかどうか等の質的要素も考慮する必要がある．

工程管理（process control）

工程管理は，開発計画に基づき，作業の進行状況を把握することが目的である．このとき，開発計画はプログラムの規模をベースとして決定することが多い．しかし，知識システムでは，見積りで述べたように，知識ベースの規模を正しく把握することが難しい．したがって，ここでも，仕様書やプロトタイプ等から知識ベースの規模を推定する方法の検討が必要となる．

品質管理（quality control）

一般に，ソフトウェアの品質評価は困難である．それは，評価のための特性要因の多様性や相反する要因の存在等による．実際には，幾つかの特性要因を組み合わせて評価基準を設定している．知識システムの場合には，知識ベースの品質評価基準を新たに検討する必要があると思われる．その特性要因としては，完全性や無矛盾性等が考えられる．なお，知識ベース以外の部分は従来システムの評価基準を適用できると思われる．

原価管理（cost management）

原価管理では，コスト要因を的確に把握し，適切な運用が行われているかチェックを行う．知識システムの場合にも，基本的には従来の技法が適用できる．ただし，原価管理はコスト見積りと関連が深いため，先に述べたコスト見積りにおける問題点が影響してくる．

製品管理（product management）

ソフトウェアを製品として管理する場合には，プログラム自身を例えば階層構造形式で管理するだけでなく，プログラムの変更を管理する必要もある．知識システムの場合には，知識ベースの変更が頻繁かつ重要であるため，その変更に対する管理方法を検討する必要がある．

### 3.6.2 知識システムの評価

完成した知識システムに対しては，各種の評価を行う必要がある．評価は，その目的に応じて試験評価，運用評価，見極め評価に分けられる．

キーワード: 試験評価，運用評価，見極め評価

試験評価 (test evaluation)

試験評価とは，システムが設計仕様を満たしているかどうかを，様々な試験を用いて評価することである．このとき，どのような知識表現方法を用いているか，あるいは推論アルゴリズムはどのようなものかといった評価対象の知識システムの持つ特徴や，ユーザインタフェースはどのような機能を持つかといったシステムの機能仕様に応じて，試験項目（何を評価するのか）及び試験方法（どのように評価するのか）を適切に選択する必要がある．試験項目としては，推論実行速度，知識検索速度等が，また，試験方法としては，ベンチマークテスト等があげられる．

運用評価 (user evaluation)

運用評価とは，システムの仕様が実際の運用状況に対して適切であったかどうかを評価することである．試験評価がシステムが仕様を満足しているかどうかの評価を行うのに対し，運用評価はシステムの仕様自身が適切であったかどうかの評価を行う．評価手順としては，まずシステムを実際にユーザや専門家に使用してもらう．次に，彼等へのインタビューを通して，システムに対する意見・要望等を引き出す．そしてそれらを分析し，システムの仕様が適切であったかどうか，また，もし問題点があれば，仕様をどのように変更・修正すればよいかを検討する．

見極め評価 (outcome analysis)

見極め評価とは，運用評価の結果を受けて，プロトタイプを実用化システムに発展させるための，あるいは運用中のシステムのバージョンアップのための総合評価を行うことである．運用評価の結果得られた各種の仕様変更・修正案に対し，コスト面等からの評価を行い，実際にシステムの仕様変更・修正を行うか行わないか，また複数

の案が存在する場合にはその優先順位の決定を行う．見極め評価の結果は，システムの設計フェーズにフィードバックされる．

### 3.6.3 知識システムの運用

知識システムの運用の際に重要なものとして，ドキュメンテーションとエンドユーザ教育があげられる．

キーワード: ドキュメンテーション，エンドユーザ教育

ドキュメンテーション (documentation)

ドキュメントはその使用者によって作成目的が異なっており，そのため，その内容も当然異なってくる．ここでは，ユーザ向け，システム開発者向け，システム運用・保守者向けのドキュメントについて述べる．

1)ユーザ向けのドキュメント

ユーザ向けには，使い方を示すために，マニュアルと呼ばれる外部仕様書が必要である．マニュアルは，さらに目的別に「入門書」・「文法書」等に分類される．そして，例えば，「入門書」には網羅的ではなくとも読みやすく理解しやすい内容が要求され，「文法書」には網羅的かつ正確な内容が要求されるといったように，各マニュアルには，その用途に応じた記載内容や記述形式が必要とされる．

2)システム開発者向けのドキュメント

システム開発者向けには，知識システムの基本仕様，機能仕様，構造仕様，論理仕様等を正確に記述した内部仕様書や，システムの開発状況を記録した開発管理・作業手順書等が必要である．

3)システム運用・保守者向けのドキュメント

システム運用・保守者向けには，知識システム内部の構造・論理の理解を助けるための内部構造説明書や，障害解析を助けるための問題解析手引書が必要である．

エンドユーザ教育 (end user education)

エンドユーザ教育の内容は，対象となる知識システムによって異なる．知識ベース

がエンドユーザに対して解放されていない場合には，システムの操作方法のみを教育すればよいのに対し，知識ベースがエンドユーザに解放されている場合には，知識の記述形式や知識ベース内の知識の体系等についても教育する必要がある．また，エンドユーザにシステムの保守・運用を任せる場合には，システムの操作方法や知識ベースに関する教育に加えて，障害時の対処方法についての教育も行う必要がある．

### 3.6.4 知識システムの保守

知識システムの保守については，まずその保守体制を確立しなければならない．また，知識ベースに関しては，その機密保持を考慮するとともに，拡張に対する手引きを用意する必要がある．

キーワード: 保守体制，機密保持，拡張の手引き

保守体制 (maintainance system)

知識システムの保守に必要な体制としては，次のものが考えられる．

1)知識ベース管理体制

知識ベースへの知識の追加・変更等に対する管理を行う体制である．知識システムの運用時には知識の追加・変更が頻繁に起こると予想されるため，その管理体制を確立しておかなければ，知識ベースの完全性が保たれなくなる．

2)機能拡張管理体制

システムの様々な機能拡張あるいは変更に対する管理を行う体制である．システム全体の統制を保つために，その拡張・変更を集中的に管理することが望ましい．

3)ドキュメント管理体制

システムの機能拡張・変更に伴うドキュメントの追加・変更等の管理を行う体制である．ドキュメントについても，その追加・変更等は集中的に管理されることが望ましい．

機密保持 (preservation of confidentiality)

知識システムにおける知識ベースは，専門家の持つノウハウの集積であるため，実

用システムとして導入する場合には，その機密保持を考慮する必要がある．特に，システムをネットワークに接続する場合には，外部から容易にアクセスできないようにプロテクトをかけたり，ユーザの監視を行うことが必要と思われる．また，スタンドアローン型のシステムについても，知識ベースの内容を容易に読み取ることができないように，知識の暗号化等の処理を施す必要があると思われる．また，電子ウイルスに対処する技術の開発も必要である．

拡張の手引き（version up manual）

知識システムは，その運用中に知識ベースの拡張が頻繁に行われると考えられる．そのため，この拡張をスムースに行うために，ドキュメントの１つとして，拡張の手引きが必要となる．これは，知識のデータ構造や知識エディタの使用法等が記述されており，通常，システム開発側が作成し，知識ベース拡張の実際の担当者（開発側の場合もユーザ側の場合もある）のもとに置かれる．

（3.6節担当　福島）

# 4．システム技術

# 4. システム技術

システム技術は，人工知能技術や知識工学技術を補完するとともに，知識システムを従来技術でつくられた既存システムに接続する上で必要とされる技術でもある．

知識システムの構築に関連するシステム技術として，

1)システム基礎技術，　2)システム化技術，　3)システム開発技術，

の３つが重要である．以下に，知識システムにおける必要性という観点から各技術の概要を示す．

## 4.1 システム基礎技術

システム基礎技術は，第１に，知識システムで取り上げる問題を分析し，ニーズ分析に基づいて知識システム的接近の必要性を明確にする上で有用である．第２に，知識処理を中心とする知識システムの限界を補完する上でシステム基礎技術は有用である．

修得しておくことが望ましいと思われるシステム基礎技術を，

①数理工学，　②システム工学，　③ソフトウエア工学，　④コミュニケーション技術，

の４つに分けて，その必要性を示す．

### 4.1.1 数理工学

キーワード：　多変量解析，多次元解析，数量化分析，最適化，信頼性，シミュレーション

あいまいなデータの中から重要な要因を発見する手掛りを与える多変量解析，多次元解析，数量化分析など確率・統計に関する手法は，知識獲得に必要なデータを整理する上で有用である．最適化，信頼性，シミュレーションなどのＯＲ的接近も知識処理の限界を補完するうえで重要な方法論である．

### 4.1.2 システム工学

キーワード：　システムズアプローチ，創造工学的手法，構造モデル，関連樹木法，
決定分析

知識システムが取り扱う悪構造・悪定義問題へ接近する上で，システムズアプローチ，すなわち，システム的かつ目標指向的な考え方とこれに基づく方法論が有用である．問題

の発見を行うための創造工学的手法（ＫＪ法・ブレーンストーミングなど）は知識獲得の初期段階で有用である．問題の階層的構造を同定する上で，構造モデル（ＩＳＭ，ＤＥＭＡＴＥＬなど）が有用である．領域専門家の評価の構造や意思決定過程を分析する上で，関連樹木法や決定分析が有用である．

4.1.3 ソフトウェア工学

キーワード： ソフトウエア開発手法，構造化プログラミング，ソフトウエア検証

ソフトウェア工学は従来型のソフトウェアの開発を支援する技術であるが，大規模知識システムの開発においては，類似する側面もあり，特に，ソフトウエア開発手法，構造化プログラミング，ソフトウエア検証については参考にすべき点も多い．

4.1.4 コミュニケーション技法

キーワード： インタビュー技法，プレゼンテーション技法，ドキュメンテーション技法

知識システムの開発は，領域専門家やユーザの協力が不可欠であり，知識システム技術者はコミュニケーション技法を修得しておくことが望まれる．特に，インタビュー技法，プレゼンテーション技法，ドキュメンテーション技法が重要である．

4.2 システム化技術

システム化技術は，知識システムの構築段階がプロトタイプ段階からフィールドテスト段階に移行するにつれて，実装との関連において重要になる．

修得しておくことが望ましいと思われるシステム化技術を，

①計算機システム，　②データベース，　③通信ネットワーク，

④ヒューマンインタフェース，

の４つに分けて，その必要性を示す．

4.2.1 計算機システム

キーワード： オペレーティングシステム，分散処理，実時間処理

知識システムを実装し，実世界を対象として実際に稼働させるためには，計算機システムについて，特に，オペレーティングシステム，分散処理，実時間処理などの技術への理解を必要とする．

4.2.2 データベース

キーワード： データモデル，データ構造，ＤＢＭＳ，ファイル編成，情報検索

知識処理技術とデータベース技術を接続・統合することにより，より柔軟な問題解決システムを実現することが期待できる．データベース技術として，データモデル，データ構造，ＤＢＭＳ，ファイル編成，情報検索などの技術を修得しておくことが望まれる．

4.2.3 通信ネットワーク

キーワード： ネットワークアーキテクチュア，ＬＡＮ，ＷＡＮ，ネットワーク管理，
最適分散化

知識システムを分散化された環境で利用することを考えた場合，通信ネットワークに関する技術が関係してくる．すなわち，ネットワークアーキテクチュア，ＬＡＮ，ＷＡＮ，ネットワーク管理，最適分散化などの技術を修得しておくことが望まれる．

4.2.4 ヒューマンインタフェース

キーワード： 人間的要因，行動分析，認知過程分析，マルチメディア，
ハイパーテキスト

知識システムは知識ベースと推論機構が完全に構築されたとしても，システムを利用するのはユーザであり，ヒューマンインタフェースを無視することはできない．理想的なヒューマンインタフェースを追及するためには，人間的要因，行動分析，認知過程分析，マルチメディア，ハイパーテキストなどに関する知見を活用することが必要である．

4.3 システム開発技術

システム開発技術は，知識システム構築のライフサイクル，すなわちシステムの計画，設計，運用，保守の各フェーズを円滑に遂行する上で参考になる．

修得しておくことが望ましいと思われるシステム開発技術を，

①システム分析と要求定義，　②システム設計，　③プログラム作成技法，

④テスト・検査，　⑤システム評価，　⑥保守・運用，

の６つに分けて，その必要性を示す．

4.3.1 システム分析と要求定義

キーワード： 要求仕様化技法、プロトタイピング、ニーズ分析、現行システム分析、
実現可能性評価

従来型システムと同様に，知識システムにおいても，システム分析と要求定義は必要である．特に，要求仕様化技法、プロトタイピング、ニーズ分析、現行システム分析、実現可能性評価などの技術はそのままでは使えないが，参考にすべき点は多々ある．

4.3.2 システム設計

キーワード： 外部設計、内部設計、データ設計、処理設計、性能設計

システム設計技法として，外部設計、内部設計、データ設計、処理設計、性能設計などは，従来型ソフトウエア向けに構成されたものであるが，知識システムの設計においても間接的に役立つ．

4.3.3 プログラム作成技法

キーワード： プログラム設計、設計記述法、プログラム言語、モジュール化

プログラム作成技法としてのプログラム設計、設計記述法、プログラム言語、モジュール化などは，従来型ソフトウエアの支援技術であり，そのまま使えるものはほとんどないが，概念的には参考になるものがある．

4.3.4 テスト・検査

キーワード： テストフェーズ、テスト技法，検査技法

知識システムにおいては，まだテスト・検査の方法が確立されてはいないが，従来型ソフトウエアにおけるテストフェーズ、テスト技法，検査技法 などの技術は知っておいた方が望ましい．

4.3.5 システム評価

キーワード： 信頼性評価，性能評価，評価技法

知識システムの評価を行う上で，信頼性評価，性能評価などの評価技法を修得しておくことは有用である．

4.3.6 保守・運用

キーワード： 評価改善，障害対策，保守・運用マニュアル，変更管理

知識システムでは，保守・運用の体制を確立しておくことが極めて重要である．そのためには，評価改善，障害対策，保守・運用マニュアル，変更管理などの技術による支援を必要とする．

以上，知識システムの構築において関連すると考えられる重要なシステム技術の概要をその必要性に重点おいて紹介した．

人工知能は知識システムによる問題解決の基礎を与える技術であり，知識工学は知識システム構築の方法論を与える技術であり，両者は知識システム構築における中心的な技術である．しかし人工知能や知識工学はまだ技術的には発展段階にあり，技術的には未成熟なところも少なくない．システム技術は人工知能や知識工学を補完する技術として有用であり，またシステム技術は従来の情報処理技術と知識処理技術の統合を図る上で必要とされる技術であることを再度強調しておきたい．

（4章担当 小林重信）

# 5．育成指針のまとめ

# 5. 育成指針のまとめ

本章では，まず知識システム技術者の多面性を指摘し，知識システム技術者の育成は，

1)システムアナリスト，　2)プロトタイピングアナリスト，

3)知識プログラマ，　4)保守・運用技術者，

の４つに分けて考えるのが適切であることを述べる．それぞれに対して，

1)守備範囲，　2)基盤技術，　3)養成方法，　4)留意事項，

を要約して，育成指針のまとめとする．

## 5.1 知識システム技術者の多面性

1.3 節において，知識システム技術者をつぎのように定義した．

「知識システム技術者とは，知識システムの研究開発プロジェクトを実施するうえで，領域専門家およびユーザの協力のもとに，システムの計画・設計・構築・運用・保守に伴う一連のタスクを実際に行うことができるものをいう」

また，1.2 節において，知識システムの開発手順をつぎのようにまとめた．

1) システムアナリシス：

①問題の設定，　②既存技術の評価，　③知識源の同定

2) プロトタイピングアナリシス：

①専門家モデルの同定，②ユーザモデルの同定

3) 知識プログラミング：

①知識表現の選択，　②知識の移植

4) 保守・運用：

①システムの評価，　②ドキュメンテーション，　③保守・拡張

このように，知識システム技術者に要請されるタスクは極めて広範囲に及び，ひとくちに知識システム技術者といっても，多様な側面をもつことにまず留意する必要がある．小規模なシステムでは，１人の知識システム技術者が，1)～4) のすべてをカバーすること

が可能であるが，システムの規模が大きくなるにつれて，役割の分担の必要性が生じてくるし，さらに大規模なシステムになれば，何人かの知識システム技術者が同一のタスクの遂行に当ることになる．

以下においては，知識システム技術者を，

1) システムアナリストとしての知識システム技術者
2) プロトタイピングアナリストとしての知識システム技術者
3) 知識プログラマとしての知識システム技術者
4) 保守・運用技術者としての知識システム技術者

の４つのタイプに分けて，各タイプの知識システム技術者の役割，必要な基盤技術や経験，養成方法などを議論することとする．

## 5.2 システムアナリストとしての知識システム技術者

### 守備範囲

システムアナリトとしての守備範囲は，①問題の設定，②既存技術の評価および③知識源の同定である．対象とする問題領域について，既存技術の限界を明らかにし，知識システム的接近の特徴を生かせるような問題であるかどうかを見極め，問題解決に必要な知識源の利用可能性を確認した上で，適切な問題を選択することが要求される．

### 基盤技術

システムアナリトとしての知識システム技術者は，システム技術，特に，システム基礎技術に精通していると同時に，人工知能の基礎と知識処理の基本を修得していることが望ましい．システム構築ツールや知識プログラミングについては常識的な理解で十分である．

### 養成方法

まず，システム的かつ目標指向的な思考を身につけさせることが必要である．従来のＯＲ技術やソフトウエア技術の特徴と限界を知り，その上で知識システムの特徴と現在の技術的な問題点を知っておく必要がある．そのためには，十分経験のある知識システム技術者の指導のもとに，ケーススタディおよび思考実験を積み重ねることにより，知識システム接近の必要条件と十分条件をチェックできるように訓練することが望まれる．

留意事項

システム技術とＡＩ技術は相補的な性質を有するが，両技術のいずれを選択すべきかを適切に判断できることが要請される．頑くなにシステム技術に固執することは無益であるし，だからといって，安易にＡＩ技術に頼るのも危険である．このことを十分理解させることが重要である．

## 5.3 プロトタイピングアナリストとしての知識システム技術者

守備範囲

プロトタイピングアナリストとしての守備範囲は，①専門家モデルの同定および②ユーザモデルの同定である．領域専門家およびユーザとの対話およびプロトタイピングを通じて，設定された問題を解決するための基本的な枠組みを明らかにすることがプロトタイピングアナリストの役割である．

基盤技術

プロトタイピングアナリストとしての知識システム技術者は，領域専門家およびユーザから必要な知識を引き出すために，インタビュー技術やプロトコル解析などのコミュニケーション技法を修得していると同時に，彼らの思考を理解するのに必要な問題領域に関する基礎知識を短期間に修得できるだけの適応性を身につけていることが望まれる．またプロトタイピングを行うために，恵まれたプログラミング環境の下にあるシステム構築ツールを一通り使いこなすだけの技量が要請される．

養成方法

プロトタイピングアナリストは，システムアナリストとしての素養と知識プログラマの素養の両方を身につけさせる必要がある．システムアナリストとしての養成方法に加えて，問題領域別のシステム構築方法論を修得させ，さらに特定の使い勝手の良いツールを使って，さまざまなタイプの問題に対して，小規模システムを早急に構築できるような訓練を行うことが望まれる．

留意事項

プロトタイピングアナリストは，人工知能や知識工学について何も知らない領域専門家やユーザを相手にするのであるから，自分の言葉ではなく，相手の言葉で，コミュニケーシ

ョンを行う必要がある．また，プロトタイピングの利点を最大限引き出すために，領域専門家やユーザからの信頼性を獲得するための努力と工夫に努めることが重要である．

## 5.4 知識プログラマとしての知識システム技術者

### 守備範囲

知識プログラマとしての守備範囲は，①知識表現の選択および②知識の移植である．プロトタイピングの分析結果およびシステムの利用形態に基づいて，適切な知識の表現と利用の方式を選択するとともに，問題解決の基本的枠組みを決定する．この枠組みのもとで，知識の抽出と知識システムへの移植ならびにユーザインタフェースの設計・製作を行うことが要請される．

### 基盤技術

知識プログラマとしての知識システム技術者は，知識工学技術，特に，構築方法論，知識獲得支援，知識プログラミングに精通していることが要請される．さらに，システム技術，特に，システム化技術についても十分な知識をもつことが望まれる．

### 養成方法

まず，LISP や Prolog などの知識プログラミングを修得させた上で，何通りかのシステム構築ツールや知識獲得支援ツール／技法を修得させることが必要である．ついで，問題領域別のシステム構築方法論をケーススタディ的に修得させる必要がある．さらに，実システムとの接続やユーザインタフェースの開発の経験を積ませることが望ましい．

### 留意事項

知識プログラマは，従来型ソフトウエア開発におけるプログラマに比べて，より広範囲の知識と技術が要請されることに留意されたい．知識プログラマは LISP や Prolog の優れた使い手でなければならないが，ハッカーであってはならない．開発される知識システムを手離れの良いシステムとするためには，プログラミング技法のセンスよりも，むしろユーザを指向した認知工学的なセンスが要請される．

## 5.5 保守・運用技術者としての知識システム技術者

### 守備範囲

保守・運用技術者としての守備範囲は，①システムの評価，②ドキュメンテーションおよび③保守・拡張である．知識システムの機能的および性能的評価について，客観的な評価体系を設定できるようになることが要請される．また，保守・拡張のためのドキュメンテーションを行うことも要請される．

基盤技術

保守・運用技術者としての知識システム技術者は，システム開発技術，特に，テスト・検査，システム評価および保守・運用に関する技術を修得しておくことが望まれる．

養成方法

従来型のソフトウエアにおける保守・運用技術者の養成方法と本質的に変るところはないが，技術的にはまだ確立されていないので，できるだけ多くのケーススタディを積ませることが必要である．

留意事項

知識システムの利点の１つは，システム開発完了後は，システム開発者の手を離れ，ユーザ自身による保守・運用に任せるところにある．したがって，ここでいう保守・運用技術者の役割は，保守・運用をユーザに任せても大丈夫なような体制を確立することにあるといえる．

以上，知識システム技術者を４つのタイプに類別して，それぞれの守備範囲，基盤技術，養成方法および留意事項を論じた．

理想を言えば，知識システム技術者の養成には，経験的な側面を必要とするため，大学の研究室における研究者の育成方法と同様な徒弟的な方式でなされることが望ましい．しかし，産業界においてそのような方式が可能なところはほんの一握りであろう．だからこそ，知識システム技術者の養成指針やカリキュラムの作成が要請されているわけであり，本報告もそのよな従うものである．本指針が知識システム技術者の養成に貢献することになれば幸いである．

（本章担当 小林重信）

平成元年３月発行

発行所　　財団法人　　日本情報処理開発協会
東京都港区芝公園３丁目５番８号
機械振興会館内
電話（０３）４３２－９３９０

６３－Ａ００４